



武汉大学

本科生课程讲义及知识总结

课程名称：空间数据库

开课学院：遥感信息工程学院

开课时间：2020-2021 年度第一学期

2020. 武汉大学

目录

第一章 · 绪论	3
➤ 时空数据库的相关概念	3
➤ 空间数据管理的发展过程	4
第二章 · 空间实体的计算机表达	6
➤ 空间实体及空间信息建模的基础知识	6
➤ 实体模型与场模型	6
➤ 空间实体的计算机表达模型	9
➤ 空间实体集的表达模型	11
第三章 · 时空信息表达	13
➤ 时空信息基准与模型	13
➤ 时空信息建模	15
➤ 时空信息表示	15
➤ 时空关系表示	17
➤ 时空数据类型	19
第四章 · 时空数据库模型	19
➤ 空间数据概念模型	19
➤ 空间数据逻辑模型	20
➤ 时态数据模型	24
➤ 时空数据模型	25
第五章 · 空间数据存储与空间索引	27
➤ 物理数据模型以及存储设施	27
➤ 文件的结构	29
➤ 索引	32
➤ 空间索引	34
➤ 时空索引	40
第六章 · 空间查询与访问	43
➤ 查询与查询语言	43
➤ 空间查询与空间 SQL	44
第七章 · 空间数据库设计	47
➤ 空间数据库设计主要内容	47
➤ 空间数据库设计过程	48
➤ 事务管理	52
第八章 · 商用空间数据库	54
➤ Oracle Spatial	54
➤ 空间数据引擎	58
➤ Geodatabase	60
第九章 · 空间数据处理与数据共享（数据获取）	60
第十章 · 空间数据库的发展	61
➤ 大数据与 NoSQL	61
➤ 空间数据挖掘	62



第一章 · 绪论

➤ 时空数据库的相关概念

1. **数据**：指客观事物的属性、数量、位置及其相互关系等的符号描述。
2. **数据与信息**：信息是经过加工的数据，数据是客观事物未经过加工的最原始状态（数据是信息的具体表现形式）
3. **空间数据**：是对现实世界中空间对象（事物）的描述，其实质是指以地球表面空间位置为参照，用来描述空间实体的位置、形状、大小及其分布特征等诸多方面信息的数据。
 - 1) **空间数据的性质**：
 - ①空间数据具有地理空间参考；②空间数据具有多尺度性，用多种比例尺表示。
 - 2) **空间数据的特征**：
 - a) **空间位置特征**：描述地理实体所在的**空间绝对位置**以及实体间存在的**空间关系的相对位置**。空间绝对位置可以通过**坐标参照系统**来描述。空间关系可以用**拓扑关系**来描述。
【拓扑关系：满足拓扑几何学原理的各空间数据间的相互位置关系】
 - b) **专题（属性）特征**：描述空间现象的**非空间特征**。它是指除了时间和空间特征以外的空间现象的其他特征，如地形的坡度、波向、某地的年降雨量等。
 - c) **时间特征**：指地理数据采集或地理现象发生的时刻或时段。
 - d) **非结构化特征**：不同于结构化的特征表达（每条记录定长，第一范式），而空间数据数据项变长，对象包含一个或多个对象，需要嵌套记录。
4. **时态数据**：随时间而变化的或具有时间语义的数据称为时态数据
5. **时空数据**：指具有时间参考的空间数据。
时空数据的特征：①时间参考；②时间多尺度性；③动态性
6. **数据库（data base）**：存储于计算机环境中的相互关联的大数据集。
◇ 数据库系统管理数据的最大优点：**数据和应用程序之间的数据独立性**。即应用程序访问数据文件时，不必知道数据文件的物理存贮结构。
7. **数据库中的数据模型**：是对数据库应用结构的结构化描述；目的是为系统开发者和用户提供一个有意义的通用计算媒介。数据模型有层次、网状、关系、面向对象四种。

8. **空间数据库**：指以特定的**信息结构**（如规划、环境、交通等）和**数据模型**（如关系模型、面向对象模型等）**表达、存储和管理**从地理空间中获取的某类**空间信息**，以满足不同用户对空间信息需求的**数据库**。
9. **使用空间数据库的目的**：
 - 1) 以最小的代价高效地存储和处理空间数据；
 - 2) 维护空间数据的现时性、一致性和完整性；
 - 3) 为用户提供现实性好，准确性高，完备，开放和易用的地理空间数据。
10. **为什么要使用空间数据库而不用数据库？**
 - 1) **与统计数据相比空间数据更复杂**：数据类型多（几何数据、关系数据、辅助数据）、数据操纵复杂（定位检索、拓扑关系检索等）、数据输出多样、数据量大，空间数据种类多。
 - 2) **空间数据的非结构化特征**。事务数据库的数据记录一般是结构化的，每一个记录有相同的结构和固定的长度，这种结构化不能满足要求空间数据存储的要求。
 - 3) **标准的 SQL 查询查询空间数据的时候会存在问题**：违背了数据独立性原则（查询的实现需要了解空间对象的结构）、方法性能低、缺乏用户友好性、不能表达几何计算等问题。
11. **时空数据库**：计算机中相互关联的有时间参考的空间数据的集合。
12. **为什么需要时空数据库？**

空间数据具有**时变性**，时空数据库有利于时空变化数据的组织、更新、查询和维护。

 - 1) 空间形态的变化：空间几何随时间发生了变化；
 - 2) 属性的变化：属性随时间发生变化

➤ 空间数据管理的发展过程

1. **人工管理阶段（20 世纪 50 年代中期）**：数据不保存，计算机主要用于科技计算，不需要将数据长期保存，需要时输入数据；没有数据管理软件；数据冗余：一组数据对应于一个程序；
2. **文件管理阶段（20 世纪 60 年代中期）**：所有数据存储在自己定义的数据结构与操纵工具的文件中；使用时给出文件名称、格式和存取方式等，其余的由文件管理系统完成。特点：数据冗余较大、程序和数据之间的独立性较差、对数据的表示和处理能力较差、数据不一致、数据联系弱。

3. **文件与关系数据库系统混合管理系统**：用两个子系统分别存储和检索空间数据与属性数据，属性数据存储在常规的 RDBMS 中；几何数据存储在空间数据管理系统中；两个子系统间用标识符联系起来（即通过关键字联系）。
✧ **1981 年 ESRI 推出第一个商用地理信息产品（ARC/INFO）**。真正有效地将地理空间技术和数据库集成于一个系统；
4. **全关系型空间数据库管理系统**：图形和属性数据都用现有的**关系数据库管理系统**管理。采用同一 DBMS 存储空间数据和属性数据——在标准的关系数据库上增加空间数据管理层；利用该层将结构查询语言（GeoSQL）转化成标准的 SQL 查询，借助索引数据的辅助关系实施空间索引操作。
5. **对象关系数据库管理系统（当今主流）**：
 - 1) **关系型数据库+空间数据引擎**（GIS 厂商研发的一种中间件解决方案）：
 - a) 用户将自己的空间数据交给独立于数据库之外的**空间数据引擎**，由空间数据引擎来组织空间数据在关系型数据库中的存储；用户需要访问数据的时候，再通知空间数据引擎，由引擎从关系型数据库中取出数据，并转化为客户可以使用的方式。
 - b) 关系型数据库是存放空间数据的容器，而**空间数据引擎则是空间数据进出该容器的转换通道**。
 - c) 典型代表：ESRI 的 ArcSDE；MapInfo 的 SpatialWare
 - 2) **扩展对象关系型数据库管理系统（数据库厂商研发，借鉴面向对象技术）**：
 - a) 系统支持定义**抽象的数据类型（ADT）**及其相关操作。用户使用增加了的空间数据类型和函数的**标准扩展型 SQL 语言**来操作空间数据，从而将空间数据类型与函数从中间件（空间数据引擎）转移到数据库管理系统中。
 - b) **抽象数据类型（Abstract Data Type, ADT）**是将数据对象、数据对象之间的关系和数据对象的基本操作封装在一起的一种表达方式。
 - c) 典型代表：Oracle 的 Oracle Spatial、IBM 的 DB2 Spatial Extender、Informix 的 Spatial DataBlade。
6. **面向对象的数据库系统**：采用面向对象方法建立的数据库系统；对问题领域进行自然的分割，以更接近人类通常思维的方式建立问题领域的模型。
✧ 面向对象数据库管理系统还不够成熟，价格昂贵

第二章 · 空间实体的计算机表达

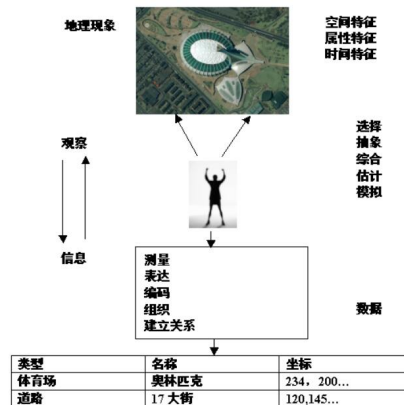
➤ 空间实体及空间信息建模的基础知识

1. 空间实体：地理空间中的物体。通常由两个部分组成：

- 1) 实体描述：实体由属性集来描述；
- 2) 空间部分（或几何部分）：包括几何和拓扑部分。

【几何指单个对象的物理位置等，拓扑指对象与对象之间的关系】

2. 空间实体的数据抽象过程：从客观世界的地理现象及其随时间的变化表示为计算机数据或信息的过程。详细过程：地理现象（空间特征、属性特征、时间特征）→人的选择、抽象、综合、估计、模拟→测量、表达、编码、组织、建立空间关系→地理数据



3. 空间信息模型：对地理空间的模型表达。

4. 空间数据模型：以计算机能够接受和处理的数据形式，为了反映空间实体的某些结构特性和行为功能，按一定的方案建立起来的数据逻辑组织方式，是对现实世界的抽象表达。

5. 空间信息建模（又称 GIS 的地理空间二分法）的分类：

- 1) 基于实体的建模 Entity-based models：把对象看作离散的，强调个体。
- 2) 基于场的建模 Field-based models：把空间看做连续的变量。

【具体见下一小节】

➤ 实体模型与场模型

1. 基于实体的空间模型的核心思想：①将地理实体和现象作为独立的对象，以独立的方式存在，主要描述不连续的地理现象；②任何现象都是一个对象，实体由不同的对象组成

2. **基于实体的空间模型的特点：**①主要描述不连续的个体现象，适合表示有固定形状的空间实体；②强调个体现象，对象之间的空间位置关系通过拓扑关系进行连接。

【基于实体的模型把信息空间看作许多对象的集合，这些对象具有自己的属性】

3. **实体模型的分类：**

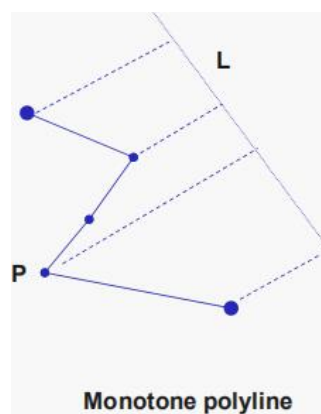
- 1) **0-维对象，又称为点 (Zero-dimensional objects or points)：**关注实体的位置而不是形状时，使用 0 维对象或点；实体面积很小，在图上空间范围缩减为一个点。

0-维对象中点对象的分类：①实体点（用来代表一个实体）；②注记点（用于定位注记）；③内点（用于负载多边形的属性，存在于多边形内）；④特征点 Vertex（表示线段和弧段上的内部点）。

- 2) **1-维对象 (One-dimensional objects)：**基本形状是折线 (polyline)；是线段或边的有限集；通常用于表示网络。两个线段的交点称为 **节点/结点** (vertex or node)；只属于一个线段的点称为**端点** (extreme points)。

1-维对象中线对象的分类：

- a) 闭合折线 closed polyline
- b) 简单折线 simple polyline/非简单折线 Non-simple polyline：没有自相交的折线成为简单折线。
- c) 单调折线 Monotone polyline/非单调折线 Non-monotone polyline：存在一条直线，从折线上的任何一个点向这条折线上引垂线都只有一个交点，这样的折线称为单调折线。



1-维对象的特性：①实体长度：从起点到终点的总长；②弯曲度：如道路拐弯时弯曲的程度；③方向性。

- 3) **2-维对象 (Two-dimensional objects)：**典型的 2D 对象表示为多边形 (polygon)；通常用于表示大面积的实体；有明确的边界。

2-维对象的分类:

- a) **简单多边形 (Simple polygon) / 非简单多边形 Non-simple polygon**: 边界是简单折线的多边形称为简单多边形。
- b) **凸多边形 (Convex polygon) / 非凸多边形 (Non-Convex Polygons)**: 任意两点之间的连线都在多边形内 (或称任意两点之间都是通视的) 的多边形称为凸多边形。
- c) **单调多边形 (Monotone polygon)**: 多边形的边界可以被分成两个单调折线的多边形称为单调多边形。
- d) **带洞多边形 (Polygon with hole)**
- e) **区域 (Region)**: 由几个多边形组成的集合。

2-维对象的特性: ①面积范围; ②周长; ③独立性或与其它地物相邻; ④内岛屿或锯齿状外形; ⑤重叠性与非重叠性。

4. 选择基于实体的建模的时候需要注意两个地方:

- 1) 尺度的选择问题: 根据应用需要选择 0-维、1-维或 2-维对象;
- 2) 线状和面状实体只是对真实实体的近似表达; 线段越多, 近似性越好; 通常在真实性和有效表达之间有一个折中。

5. 场模型的核心思想: 把地理空间的事物和现象作为连续的变量来看待; 在空间中任何点上都有一个表达这一现象的值, 对连续的地理现象进行建模。

6. 场模型的表达类别 (根据属性分布的表示方法):

- 1) **图斑模型**: 将地理空间划分为一些简单的连通域, 每个区域用一个简单的数学函数表示一种主要属性的变化。

图斑模型对应的属性函数的分类:

- a) **常量**: 属性函数值是一个常数, 如土壤类型;
 - b) **线性函数**: 属性值的变化是一个线性函数;
 - c) **高阶函数**: 要求属性函数是一个高阶函数, 以提高表示精确性。
- 2) **等值线模型**: 场由一系列等值线组成。
- ✧ 等值线: 地面上所有具有相同属性值的点的有序集合。
- 3) **选样模型**: 地理空间上的属性值是通过采集有限个点的属性值来确定的。

采样方法:

- a) **离散点**: 表示区域 D 上的三维向量有限序列, 用函数的形式描述。
 - b) **规则格网**: 通常是正方形, 也可以是矩形、三角形等规则网格。规则网格将区域空间切分为规则的格网单元, 每个格网单元对应一个数值。
 - c) **不规则三角网采样**: 按照一定的规则将离散点连接成覆盖整个区域且互不重叠、结构最佳的三角形。
7. 场模型和实体模型 **可以共存**。
8. 场模型和实体模型的建立相当于数据库的概念模型的设计。

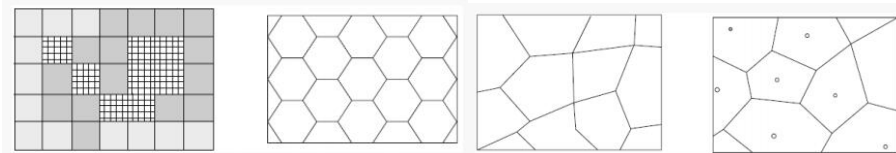
➤ 空间实体的计算机表达模型

1. 空间实体的计算机表达模型 (数据结构):

- 1) **镶嵌模型 (曲面细分模型)**: 用一个离散的空间来近似的表达连续空间;
- 2) **矢量模型、半平面模型**: 构建数据结构表达连续空间。

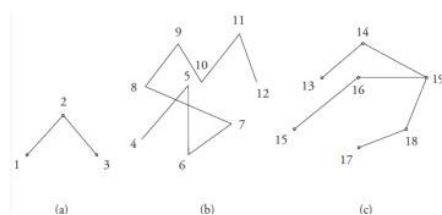
2. **镶嵌 (曲面细分) 模型 Tessellation Mode**: 借助格网, 将连续的空间分解为一系列离散化的单元格 (cell)。在镶嵌模型中, 根据单元格的形状可以分为 **规则镶嵌模型 (Regular Tessellation)** 和 **不规则镶嵌模型 (Irregular Tessellation)** 两种。

◇ 不规则镶嵌模型的单元格的形状和大小可变。



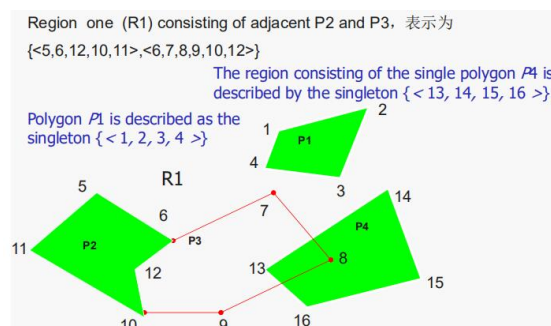
3. **用镶嵌模型表达基于实体模型的对象**: 二维空间中的空间实体被表示为 **包含它的有限像素子集**。
- 1) **分类**: ①点实体: 被描述为一个像元, 用像元的地址来代表点的位置; ②折线、多边形、区域: 用有限个像元构成的像元集来表达。
 - 2) **镶嵌模型中空间实体的最小表达单位**: **一个单元或像素 (Cell 或 Pixel)**, 依行、列构成的单元矩阵叫 **栅格 (Grid)**。
 - 3) 镶嵌模型表示法的 **精度与分辨率有关**。
4. **用镶嵌模式表达基于场模型的对象**: 基于场模型的数据在镶嵌模型中, 不再是点的连续函数, 而是 **像素的连续函数**。
- ◇ **不规则三角网 (Triangulated Irregular Network, TIN)** 属于镶嵌模型的一种, 单个三角形的 **顶点就是原始数据点或其它空间信息的控制点**。

5. **嵌入式空间**: 空间对象存在于“空间”之间称为嵌入式空间，空间对象的定义取决于嵌入式空间的结构。常用的嵌入式空间有：欧氏空间（距离和方位）、量度空间（距离）、拓扑空间（拓扑关系）、面向集合的空间（集合关系（包含、合并等））
 - ✧ 将地理要素嵌入到欧氏空间中，分为三种基本对象：点（最基本元素）、线、面；
6. **矢量模型 Vector Mode**: 定义一种数据结构来描述连续的空间。在矢量模型中，对象的几何特征是用点和边这样一些基元/图元来构造的，线是点的序列；相对于栅格表达，矢量表达不占据太多的内存。
7. **矢量模型表达空间实体数据 (Entity-based data in Vector model)**:
 - 1) **表示方法**:
 - a) **point** : [x: real, y: real];
 - b) **polyline** : < point >: 折线表示为点的列表。
 - c) **polygon** : < point >: 多边形也表示为点的列表，区别在于 polygon 是一个闭合的 polyline, 点对 (pn, p1) 也是 polygon 的一条边。
 - d) **region** : { polygon }。
 - ✧ 元组 (tuples) 用 [] 表示；列表 (lists) 用 <>；集合 (sets) 用 {}。
 - 2) **需要注意的地方**:
 - a) 每一个多边形都可能有 2n 个可能的表达方式 (n 为节点数)；
 - b) 线结构和多边形结构没有明显的区别，在实际表示中可以用标识来区分。
 - 3) **矢量模型表达空间实体数据的实例**:



$L1 = \langle 1, 2, 3 \rangle$; $\square L2 = \langle 4, 5, 6, 7, 8, 9, 10, 11, 12 \rangle$;

$L3 = \{ \langle 13, 14, 19 \rangle, \langle 15, 16, 19 \rangle, \langle 17, 18, 19 \rangle \}$



- 4) 矢量模型表达空间实体数据的特点：①矢量模式比栅格形式更紧凑。②表达空间实体时，在表达能力和结构的复杂度上有一个折中。③简单的矢量表达不能表示一些约束，而这些约束对确保语义的正确和数据完整性方面是很有用的。

8. 矢量模型表达场数据 (Field-based data in Vector model):

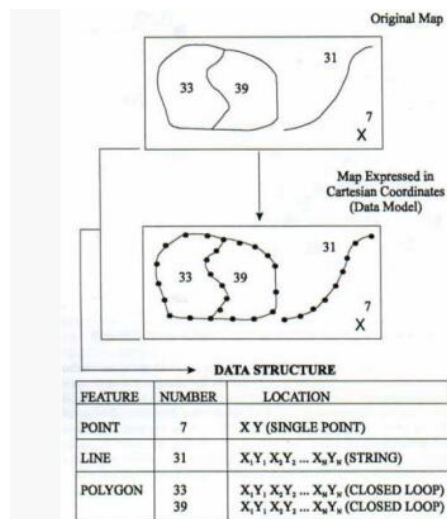
- 1) 基本思想：首先需要对基于场模型建模的数据利用镶嵌模型得到其近似的表示，然后利用矢量模型的数据结构对镶嵌模型的单元格进行表示。
- 2) 典型例子：TIN 是一种场模型，本身属于镶嵌模型，但可以用矢量模型来描述。

➤ 空间实体集的表达模型

1. 空间关系 (spatial relationships): 又称为实体和实体之间的关系。
2. 空间实体集的表达模型可分成三种：面条 Spaghetti、网络 Network、拓扑 Topological
3. 面条模型 Spaghetti Model:

- 1) 特点：①独立地描述实体集中任意实体的几何信息；②面条模型不存储拓扑关系，所有的拓扑关系在需要的时候进行计算；③没有数据共享，公共边被重复记录，同一个点可能数据不一致，数据冗余大；④不能方便的表达带有“岛”“洞”的复杂多边形。

- 2) 面条模型的表示方法:



4. 拓扑模型 (Topological Model):

【GIS 只记录表达了两类拓扑关系：拓扑邻接和拓扑连通关系】

- 1) 特点：①不仅记录对象的几何特征，同时还记录对象间的空间关系；②拓数据模型可以表达相邻的多边形，拓扑关系不需要实时计算；③几何数据是共享的，共

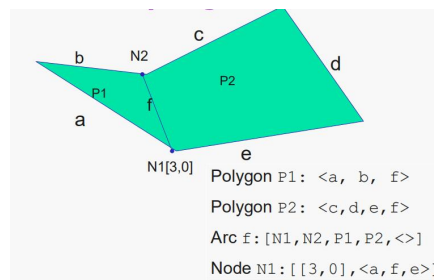
用边只需要更新一遍；④有利于网络和路径的分析；⑤一些结构信息没有实际的语义含义；⑥增加一个新的对象需要对平面图重新进行计算。

- 2) 拓扑模型引入了节点 (node) 和弧 (arc) 的定义：①弧是连接起始节点和终止节点的线，而节点是弧的两个端点（区分于实体/常规点）；②弧的起点和终点定义弧的方向；③多边形被表示为弧的列表，每条弧为相邻多边形所共享，区域被表示为一个或一组多边形。

☆ 注意：节点一定是常规点，但常规点不一定是节点。

- 3) 拓扑模型的表达方式：

- a) point : [x:real, y:real]
- b) node: [point, <arc>]
- c) arc: [node-start, node-end, left- polygon right-polygon <point>]
- d) polygon: <arc>
- e) region: {polygon}



5. 网络模型 (Network Model): 主要用于一些线性网络应用，基于数学中的图论，只存储点和线的拓扑关系，是拓扑模型的特例。

- 1) 网络模型的表达方式：

- a) point : [x:real, y:real]
- b) node: [point, <arc>]
- c) arc: [node-start, node-end, <point>]

- 2) 网络模型的种类：①Planar, 每个边的交点记录为一个节点 (node), 即使这个节点不对应于任何地理实体；②Noplanar, 边的交叉不产生交点。

- 3) 网络模型的特点

- a) 优点：网状模型可以表示多对多的关系，其数据存储效率高于层次模型。
- b) 缺点：①没有 2-维对象之间的拓扑关系，只有点和线对象之间的拓扑关系；②网状结构复杂，增加了用户查询和定位的困难。

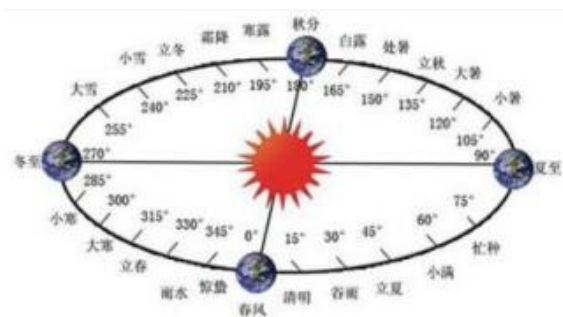
第三章 · 时空信息表达

➤ 时空信息基准与模型

1. **时间**：时间是一种只能从其对自然变化的影响中才能觉察到的现象。用以描述客观现象发生变化的顺序，与空间不可分割地联系在一起，并通过空间表现出来。
2. **时刻和时间间隔（时间的含义）**：
 - 1) **时刻**：表示事物在运动过程中的某一瞬间，即某一事物或现象发生的瞬间。
 - 2) **时间间隔**：也称为时间段，指某一事物或现象发生所经历的过程，是这一过程始末的时刻之差。
3. **时间的特性**：一维性、客观存在性、通用性、连续性、可量测性、单向性。
4. **时间的表现形式**：
 - 1) 线性结构：



- 2) 循环结构：



- 3) 分支结构：



5. **常用时间系统**：
 - 1) **世界时 (UT, Universal Time)**：格林尼治平时 T (0 时区的区时) 称为世界时。
世界时是以地球自转周期为基准测定的。
 - 2) **历书时 (ET, Ephemeris Time)**：地球绕日公转周期为基础，历书时的基本单位采用国际度量衡委员会所定义的秒长。

3) **原子时 (AT, Atomic Time)**: 用原子跃迁所产生的振荡频率为基准而建立起来的一种均匀的计时系统。原子时直接定义秒长, 属物理时。秒是相当于铯原子 133 在两个基态的超精细结构的能级跃迁辐射的电磁振荡 9192631770 周所经历的时间。

4) **协调时 (UTC, Coordinated UT)**: 是介于原子时与世界时之间的一种均匀时。它以原子时为基准, 即以原子时的秒长作为计量时间的基本单位, 在时刻上进行调整, 使其与世界时 UT1 时刻之差不超过 $\pm 0.9s$, 这样确定的一种时间系统, 称为协调世界时, 简称协调时, 用 UTC 表示。

6. 时间的尺度:

1) 测量时间, 必须建立一个测量基准, 即**时间的单位 (尺度) 和原点 (起始历元)**; 时间单位, 作为时间的尺度基准是关键; 原点, 作为时间系统的起算时刻是可以根据实际应用加以选定。

2) **时间度量基准的建立需要依靠事物 (物体) 或现象在一定条件下的运动为基础。**

a) **选择的运动的条件**: 运动是连续、周期性的; 运动周期充分稳定; 运动周期必须具有复现性。

b) **主要的时间系统所依赖的运动**: 地球自转, 是建立世界时基准的基础; 行星绕太阳的公转运动 (开普勒运动), 是建立力学时基准的基础; 原子谐波振荡, 是建立原子时的基准; 脉冲星发射周期性脉冲信号, 是建立脉冲星时的基准。

3) **时间粒度**: 具有相同长度的时间段称之为时间粒度 (granularity), 也称为时间分辨率或时间标度。选用不同的时间粒度, 表明了**时间的多尺度性**。

4) **时间尺度的不确定性**: 指某事件发生是已知的, 但何时发生时未知的, 则称该事件是时态非确定的。

造成不确定性的原因:

a) **预测的不精确性**, 绝大多数系统的预测时间是不精确的。

b) **事件时间的不确定性**, 有时实际事件发生的时间是不确定的日期。

c) **微粒过小**, 例如数据库里计时单位为秒, 而实际事件是以天来计算的。

d) **计时的不确定性**, 即使数据库的计时单位和实际事件发生的时间相一致, 但大多数计时设备是不精确的。

➤ 时空信息建模

1. 状态、事件与时段、时刻的关系：

- 1) 一个对象在其生命周期 (life-span) 里有不同的状态，事件是对象从一个状态到另一个状态的质变过程，而状态可以认为是对象逐渐进化的过程。
- 2) 事件的时间采用时刻来表示： $T_i = T_0 + S \times n$
- 3) 状态的时间采用时间段（时段）表示： $\Delta T = |T_k - T_l|$

2. 基于事件的时空建模：

$$E_i = [ID_{Ei}, G_i(x, y), T_i]$$

ID_{Ei} ：事件的标识； $G_i(x, y)$ 是第*i*个事件发生后对象的空间状态； T_i 事件发生的时刻

3. 基于场的时空模型：

从对象运动的角度，状态随时间的变化具有渐进性，是一个连续的过程，对于连续过程的建模，通常都采用场模型。定义其场模型需要确立时间框架（Time framework）、场函数（field function）和一组相关的场操作（field operation）三个组成部分。

$$f_i: TF \rightarrow G_i$$

f_i ：场函数：一个将时间框架映射到不同空间属性中的函数

TF ：时间框架：以一定时间基准为基础而建立的时段

G_i ：空间特征域

✧ 时空场模型与空间场模型所不同的是，空间场模型是将离散的空间框架映射到属性域，而时空场是将离散的时间框架映射到空间域。

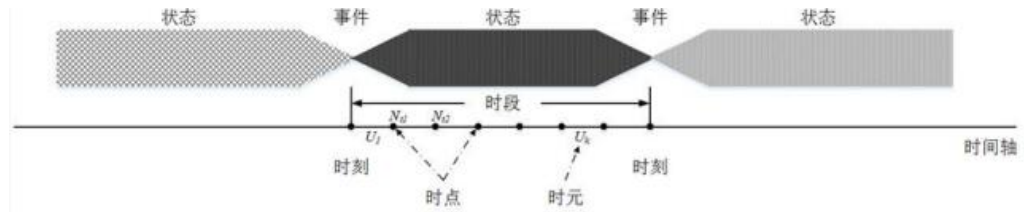
➤ 时空信息表示

1. 时空信息建模是在较高的抽象层次讨论如何对客观世界的时空现象的运动过程进行表示，在这个层次中，地理空间现象的运动过程看成渐变和跃变。时空信息表示主要研究地理对象的时空运动过程在计算机中的表达问题，包括时间的表示、离散事件的表示以及状态的表示。
2. 时间表示的两种方式：①离散的（discrete），类似自然数，非负整数的集合，每个时刻之后都有一个后继者。②连续的（continuous），类似于实数，任意两个时间点 t_1 、 t_2 之间总能找到第三个时间点 t_3 。
3. 离散事件的表示：

事件是对在某一时刻发生质变的空间现象的建模，事件记录对象在一个时间点的空间状态。事件[对象几何特征（空间特征），空间特征（时刻）]。

4. 状态的表示：

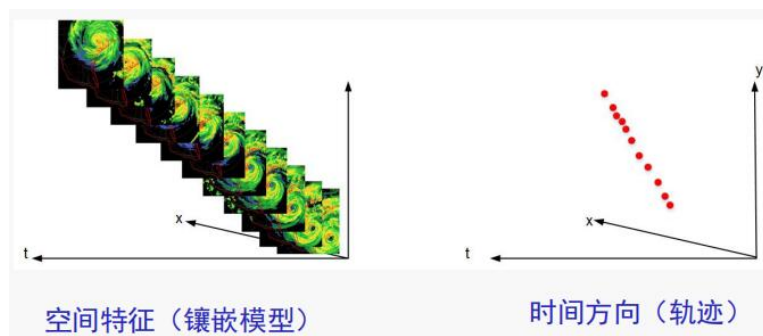
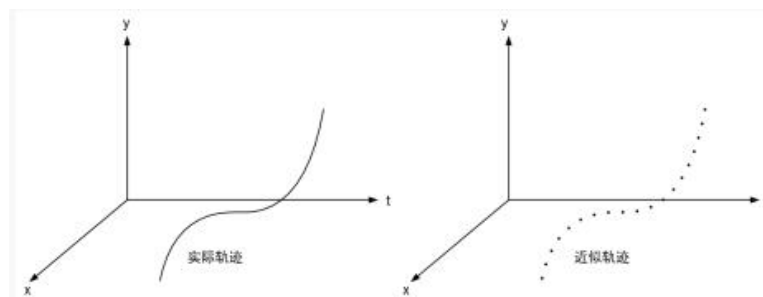
状态是对象或连续地理现象随时间渐进变化过程的表示，是对持续过程的建模。



时点：是构成时元的端点，通常也是一个时间点或时刻。

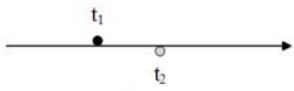
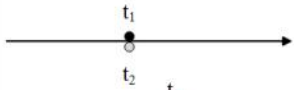
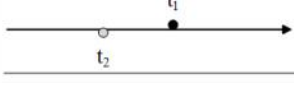
状态的时间段首先离散化为时元，记录时点的特征，通过离散的时点的状态近似时段上的时空过程

对象的位置随时间改变，形状、大小不发生变化的状态表示方式：时空过程表现为一个对象的位置随时间移动后所形成的一个连续轨迹，对其时段进行离散后则可得到用于逼近或近似对象运动过程的离散轨迹。

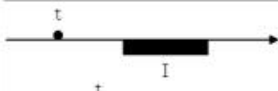






➤ 时空关系表示


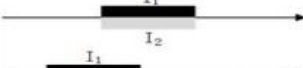

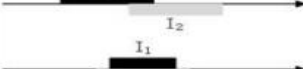
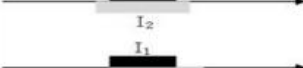
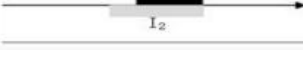
1. 为什么要研究拓扑关系：用拓扑关系表示，不论怎么变化，其邻接、关联、包含等关系都不改变。拓扑关系能够从质的方面和整体的概念上反映空间实体的空间结构关系
2. 空间实体的基本拓扑关系 (3种)：拓扑邻接关系、拓扑关联关系和拓扑包含关系。
 - 1) 拓扑邻接和拓扑关联是用来描述结构元素（比如结点、弧段、面域）之间的两类二元关系。
 - 2) 拓扑邻接关系存在于同类型元素之间（注意是“偶对集合”）。一般用来描述面域邻接。
 - 3) 拓扑关联关系存在于不同类型元素之间。常用来描述结点与边、边与面的关系。
 - 4) 拓扑包含关系用来说明面域包含于其中的点、弧段、面域的对应关系。包含关系有同类的，也有不同类的。
3. 时态拓扑关系：时刻-时刻之间，时刻-时段之间，时段-时段之间则存在顺序相关的关系，将这种关系称为时态拓扑关系。
4. 时刻-时刻拓扑关系：

关系	算子	关系描述
	$t_1 \text{ before } t_2$	t_1 先于 t_2
	$t_1 \text{ equals } t_2$	t_1 和 t_2 同时
	$t_1 \text{ after } t_2$	t_1 后于 t_2

时刻-时段拓扑关系：一个时间点和时间段之间的关系。设 T 表示时间的集合， $t \in T$ 表示时间集合中的时刻， $I \in T$ 表示时间集合 T 中的一个时间段。

























































关系	算子	关系描述
	$t \text{ before } I$	t 先于 I
	$t \text{ starts } I$	t 终于 I 起点
	$t \text{ during } I$	t 终于 I 期间
	$t \text{ finish } I$	t 终于 I 终点
	$t \text{ after } I$	t 后于 I

6. 时段-时段拓扑关系：假设 T 表示时间的集合， $I_i \in T$ 表示时间集合 T 中第 i 个时间段

关系	算子	关系描述
	$I_1 \text{ before } I_2$	I_1 先于 I_2
	$I_1 \text{ equal } I_2$	I_1 和 I_2 同时
	$I_1 \text{ meet } I_2$	I_2 开始于 I_1 结束
	$I_1 \text{ overlap } I_2$	I_1 和 I_2 重叠，且先于 I_2 开始
	$I_1 \text{ during } I_2$	I_1 在 I_2 同期间
	$I_1 \text{ start } I_2$	I_1 和 I_2 同时开始，且 I_2 先于 I_1
	$I_1 \text{ finish } I_2$	I_1 和 I_2 同时结束，且 I_2 晚于 I_1

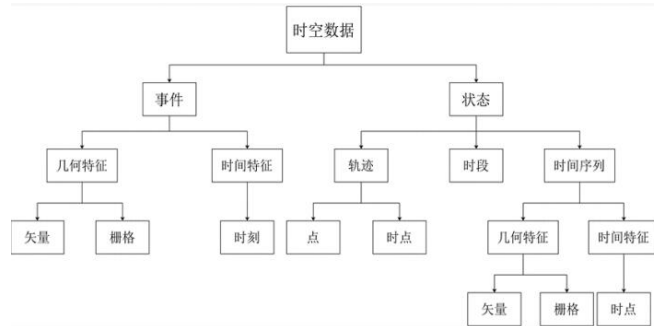
7. 时空拓扑关系：

- 1) 时空关系：结合时间维度和空间维度，表达在一个时空域内两个区域间的时空关系；
- 2) 时空域的构造：将二维空间内的二元关系映射（非投影）到相关的一维空间中的二元关系上，因此时空域的第一维度则由空间关系给出，第二维度由时间给出。
- 3) 最小时空关系组合中获得的时空域中八个 TSR 最小正交关系，在表中一个关系由一个三元组 (SR, TR, TSR) 所构成，表中定义了 71 种关系，其中，8 种空间关系 (SR)、7 种时态关系 (TR) 和 56 种时空关系 (TSR)。

SR	TR	equals	before/after	meets/met	overlaps/overlapped	during/contain	starts/started	finishes/finished
equals		 EQUAL	 DISJ	 TOUCH	 OVLP	 CVRD/CVR	 CVRD/CVR	 CVRD/CVR
touch		 TOUCH	 DISJ	 TOUCH	 TOUCH	 TOUCH	 TOUCH	 TOUCH
in		 CVRD	 DISJ	 TOUCH	 OVLP	 IN/OVLP	 CVRD/OVLP	 CVRD/OVLP
contain		 CVR	 DISJ	 TOUCH	 OVLP	 OVLP/CON	 OVLP/CVR	 OVLP/CVR
cover		 CVR	 DISJ	 TOUCH	 OVLP	 OVLP/CVR	 OVLP/CVR	 OVLP/CVR
covered		 CVRD	 DISJ	 TOUCH	 OVLP	 CVRD/OVLP	 CVRD/OVLP	 CVRD/OVLP
overlap		 OVLP	 DISJ	 TOUCH	 OVLP	 OVLP	 OVLP	 OVLP
disjoint		 DISJ	 DISJ	 DISJ	 DISJ	 DISJ	 DISJ	 DISJ

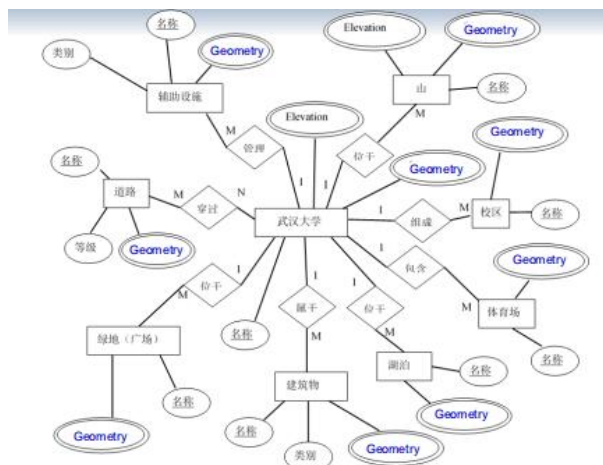
➤ 时空数据类型

1. 时空数据是由具有时间参考的地理空间数据和专题属性数据构成的，即，由空间数据和时间数据构成。

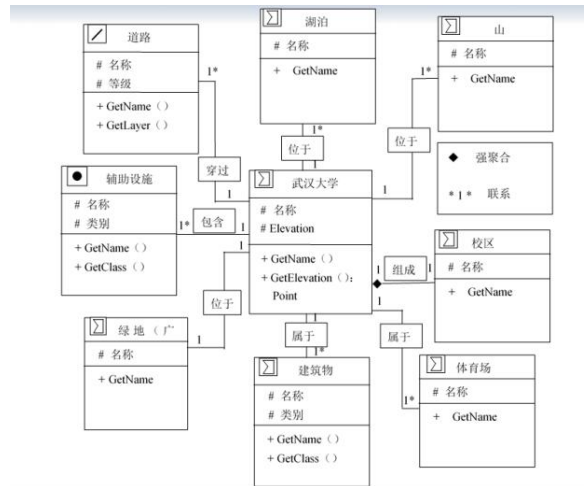


➤ 空间数据概念模型

1. **为什么要进行空间数据组织：**为了简化庞杂的对象（所有的地物类）或为了显示、制图、存取、查询的方便，对庞杂的对象分层、分幅和分块组织。**空间数据组织有空间数据分层和空间数据分幅。**
2. **概念模型：**指概念化的数据模型，是在**建模初始阶段形成的对真实世界的认识**，也是**数据库设计中数据建模的第一步**，概念模型中描述的内容会被进一步转化为数据库支持的逻辑数据模型。
3. **概念模型的建模常用工具：****实体关系（ER）模型、统一建模语言（UML）。**
 - 1) **空间数据的 ER 模型：**E-R（实体-联系）模型把要描述的对象抽象为实体和实体之间的联系，根据联系的类别将实体之间的关系表达出来，是最常用的概念模型之一。E-R 模型用**图形化**的方法表示概念模型。



- 2) **UML 模型**：是用于面向对象软件设计的概念层建模的新兴标准之一，用于在概念层对结构化模式和动态行为进行建模。将 UML 的静态建模用于数据库设计中，使用类图来对数据库建立概念模型。



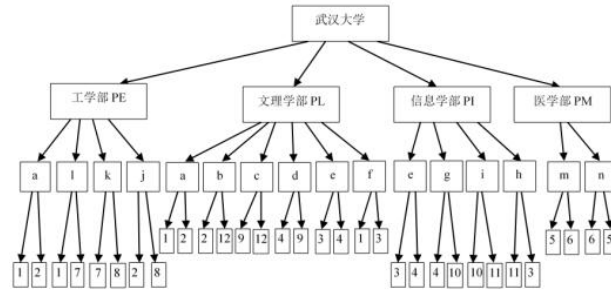
模型的相关定义：

- 类：类是指应用中具有相同性质的对象的封装，等价于 ER 模型中的实体；
- 属性：属性描述类的对象；
- 方法：方法指一些函数，是类定义的一部分，用于修改类的行为或状态；
- 关系：关系是一个类与另一个类或者与它自己的联系

关系有聚合、泛化、关联等。

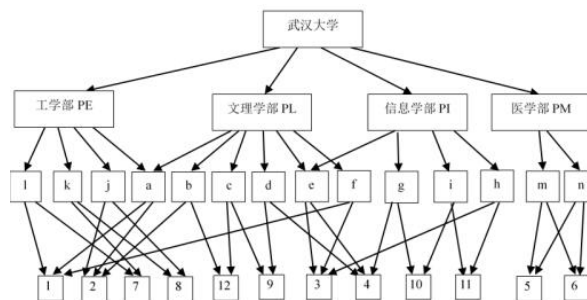
➤ 空间数据逻辑模型

- 数据库的逻辑模型（在数据库中又称数据模型）**：指数据及其联系的逻辑组织形式的表示，是数据库的核心问题。通常的数据模型包括：**数据结构、数据操作和数据的完整性约束**。
- 空间数据逻辑模型**：描述空间数据库的数据内容和结构，是 GIS 对地理数据表示的逻辑结构，属于数据抽象的中间层，由概念数据模型转换而来。
- 经典数据库系统中常用的 3 种数据模型有**：**层次数据模型、网状数据模型、关系数据模型**。将这三种模型应用到空间数据的组织上形成：**层次空间数据模型、网状空间数据模型和关系空间数据模型**。
- 层次空间数据模型**：用树形结构表示实体和实体之间的联系。
 - ◇ 层次模型结构清晰、层次分明，容易实现，能够很好地反映出层次特征



网状空间数据模型：用网络结构表示实体和实体之间的联系

◇ 基本特征是节点数据之间没有明显的从属关系，表现为有向图结构，实质上是若干层次结构的并集，能反映现实中常见的多对多关系。



关系空间数据模型：将数据的逻辑结构用满足一定条件的**二维表**来表示，二维表成为“关系”。二维表表示**同类实体的各种属性的集合**，每个实体对应其中的一行，在关系中叫做**元组**，相当于一个**记录**。

多边形号	边号	边长
PE	a	62
PE	l	19
PE	k	48
PE	j	18
PL	a	62
PL	b	41
PL	c	28
PL	d	20
PL	e	22
PL	f	23
PI	e	22
PI	g	22
PI	i	31
PI	h	35
PM	m	36
PM	n	60

边号	起始点号	终止点号
a	1	2
b	2	12
c	12	9
d	9	4
e	4	3
f	3	1
g	4	10
h	11	3
i	10	11
j	8	2
k	7	8
l	1	7
m	6	5
n	5	6

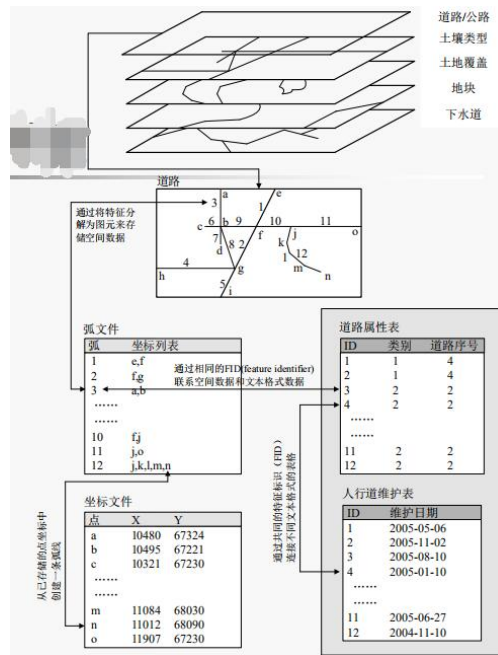
编号	x	y
1	x ₁	y ₁
2	x ₂	y ₂
3	x ₃	y ₃
4	x ₄	y ₄
5	x ₅	y ₅
6	x ₆	y ₆
7	x ₇	y ₇
8	x ₈	y ₈
9	x ₉	y ₉
10	x ₁₀	y ₁₀
11	x ₁₁	y ₁₁
12	x ₁₂	y ₁₂

地理关系空间数据模型：用**文件形式**存储地理数据中的空间数据及其拓扑关系数据，

利用**关系数据库 RDBMS** 的表存储属性数据，通过**唯一标识符**建立它们之间的关联。

空间数据被抽象成一系列独立定义的**层**，每层代表了一个**相关空间要素的集合**。

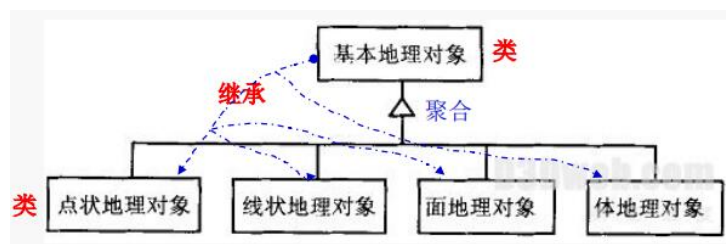
- 1) **特点：**
- ① 空间数据和属性数据相结合。空间几何数据和属性数据采用二元存储，几何空间数据存放在建立了索引的二进制文件中，属性数据存放在 DBMS 表里面，二者通过标识符进行连接。
 - ② 可以存储实体对象要素的拓扑关系。



应用：ESRI 的 ShapeFile 模型、Coverage 使用的 Geo-Relation Model

- ShapeFile 模型**：.shp (坐标文件)，.shx (索引文件)，.dbf (属性文件)
- Geo-Relation Model**：TIC 文件 (即地面控制点文件)，BND 文件 (控制一个 Coverage 的范围文件)，ARC 文件 (弧段文件)，PAT 文件 (属性文件)，LAB 文件 (多边形内部标识点文件)。
- Coverage 主要特点**：空间数据与属性数据相结合 (二进制文件+DBMS 表)；能够存储矢量要素之间的拓扑关系。

8. **面向对象的空间数据库模型**：面向对象数据模型就是一个由类及类的继承与合成关系所构成的类层次结构图。例如：



- 对象**：由一组属性、一组方法和一个对象标识符 OID 所构成的一个封装体。每个概念实体都建模为一个对象，可以描述为一个三元组： $object = (ID, S, M)$ 。
- 类**：具有相同属性和相同方法的一组对象的集合。对象是类的一个实例，例如 $class = (CID, CS, CM)$ ，其中 CID 是类标识，CS 是类的状态描述，CM 为类的操作。

- 3) **类层次结构**: 由类与类之间的继承关系和合成关系所构成的一个层次关系图。
- 4) **继承**: 父类和子类之间共享数据结构和方法的机制, 是类之间的一种关系。
- 5) **关联**: 在现实中, 对象之间会发生某种联系, 程序世界中用“关联”来表示。
- 6) **聚集**: 聚集是对象之间的另外一种关联, 表示类之间是整体与部分的关系。
- 7) **组成**: 聚集对象和它的组成对象之间有强关联时, 我们把这种聚集叫做组成。在组成关系中, 整体拥有各部分, 部分与整体共存。

9. 空间数据的“几何”对象模型:

图元: 在几何对象模型中的许多 basic 几何类型, 用于构建几何对象。

对象与图元的关系: 一个对象由一个或多个图元来构建 (一个: 简单对象; 多个: 复杂对象)。

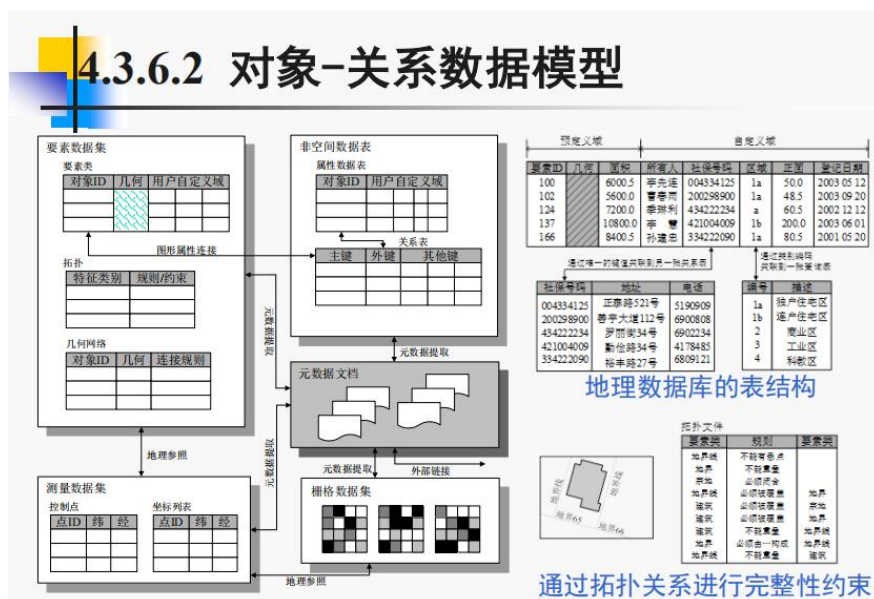
10. 对象关系空间数据模型: 在一个数据库内同时存储、查找和管理空间数据的复杂关

系, 既可以利用关系型数据库管理系统完善的数据管理功能, 又可以使用面向对象技术的功能, 方便地管理空间数据的复杂关系。

【无论是空间还是属性都用关系型数据库进行存储, 可以简单地理解为放在一个表】

◇ 对象关系数据模型利用对象技术**对关系模型进行扩展**, 即对关系数据库的数据类型进行了扩展 (增加管理空间数据的数据类型)。扩展类型有: 大对象 LOB(Large Object)类型、BOOLEAN 类型、集合类型(Collection Type)ARRAY 等。

◇ 传统关系模型称为“平面关系模型”, **不允许表中有表**。对象关系数据模型在原有关系模型的基础上**增加了元组、数组、集合等数据类型**。





➤ 时态数据模型

1. **时态属性**：和时间相关的属性称为时态属性，属性可以有两种形式和时间相关联：

- 1) 属性取时间值，即其值域为时间；
- 2) 属性的值是关于时间变化的。

2. **时态数据库的时间体系**：

- 1) **用户自定义时间**：用户自定义时间指用户根据自己的需要或理解定义的时间，可以在数据表建立或结构修改时，如同其他标准数据类型一样为用户所用。
- 2) **有效时间**：指一个对象在现实世界中发生并保持的那段时间，或者在现实世界中为真的那段时间。（含义依赖于具体应用，取值是否有效视具体应用场合而定，有效时间由数据库系统解释；可以反应过去、现在和将来的时间，可以被更新）

✧ **有效时间的判断**：

如果(*StartTime*, *EndTime*)是事件的有效时间，规定变量*Now*为历史库的当前时刻，如果 $EndTime \geq Now \geq StartTime$ ，则表示所述事件仍然合法。如果 $EndTime < Now$ ，则表示该事件已成历史，不再有效。

- 3) **事务时间**：指一个数据库对象进行操作的时间，记录着对数据库修改或是更新的各种操作历史。（事务时间不能晚于现在时间，因为它反映着数据库实际操作时间，即不能指向未来时间）

3. **时间概念模型**：是对真实世界中时间域的概念化建模，是时间域的概念设计。

4. **时态数据库逻辑模型**：

- 1) **历史关系数据模型**：引入“存在期 (lifespan)”表示时间维。三个粒度：
 - a) 数据库关联一个存在期：每个关系和关系中的每个元组都具有相同的存在期
 - b) 每个关系与存在期关联：每个关系可以定义不同的时间段，但对于一个给定关系中每个元组在时间维度上是一致的
 - c) 每个元组与存在期 (lifespan) 关联：每个元组可以定义不同的时间，但一个元组的同一属性时间维度上是一致的。
- 2) **数据模型对象**：在数据模型对象中扩展时间来实现时间方面的管理
 - a) **时间点数据模型**：用时刻或时间点来标记元组的时间戳。用一个元组表示一个在每个时间点有效的事实
 - b) **时间段（间隔）数据模型**：使用时间段作为时间戳。每个事实与事实的有效

时间相关联。

- c) 时间元数据模型：时间元（temporal elements）是有限的时间段的联合。
 - d) 属性值时间戳模型：为了在单个元组中表示关于真实对象的所有信息，引入了属性值时间戳模型。
 - 3) **快照数据模型**：快照数据模型是以特定时刻的瞬间快照为模型来反映现实世界，不支持有效时间和事务时间，只支持用户自定义时间。
 - 4) **回滚数据模型**：回滚数据模型按照事务时间编址，支持保存过去每次事务提交，状态变迁之前的数据。（属性维+元组维+事务时间维）（过去元组的错误决不可以更正，而只能查看）
 - 5) **历史数据模型**：就是支持有效时间的数据模型。（属性维+元组维+有效时间维）
 - 6) **双时态数据模型**：既支持事务时间又支持有效时间，储存了数据库和现实世界两者发展的历史
5. **历史数据模型 vs 快照数据模型**：
- 1) 相似点：
 - a) 历史数据模型是快照数据模型的扩展；
 - b) 历史数据模型和快照数据模型一样，可以任意修改以前的状态；
 - c) 历史数据模型和快照数据模型一样，不保留对数据修改的中间状态。
 - 2) 不同点：历史数据模型支持有效时间，而快照数据模型只支持用户自定义时间。
6. **历史数据模型 vs 回滚数据模型**：①二者都是三维结构，二者都对时态有一定的支持。②历史数据模型支持有效时间，而回滚数据模型支持事务时间。

➤ 时空数据模型

- 1. **时空数据的基本特征**：时间、空间和属性和动态（时变）
- 2. **时空变化类型**：
 - 1) 连续变化：当某对象或现象在时间上不间断变化，表示从起点到终点的连续变化
 - 2) 周期变化：当地学现象以规则或可预期的频率变化时，称为周期变化；
 - 3) 间歇变化：在时空上无规律的间歇变化。
- 3. **时空数据模型（时空一体化数据模型）**是时空信息系统及时空地学可视化的基础，能有效组织和管理时态地学数据，是一种属性、空间和时间的语义更完整的地学数据模型。

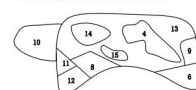
4. **序列快照模型 (sequent snapshots)**: 将某一时间段内地理现象的变化过程, 用一系列时间片段的序列快照保存起来, 反映整个空间特征的状态, 根据需要对指定时间片段的现实片段进行播放, **快照间的时间间隔不一定相同**。(属性维+元组维)
5. **基态修正模型 (base state with amendments)** 也称之为底图叠加模式, 它按事先设定的时间间隔采样, 不存储研究区域中每个状态的全部信息, 只存储某个时间的数据状态, 以及相对于基态的变化量, 避免连续快照模型将每张未发生变化部分的快照特征重复进行记录。【相对序列快照数据量大大减小, 提高了时态分辨率】
6. **离散网格单元列表模型**: 将网格单元及其变化以变长列表形式存储。每个**网格单元**列表的一个元素对应于该位置上的一次时空变化。
7. **时空复合模型**: 时空复合模型将系统空间分割为若干个时空单元, 若时空单元发生分裂, 则用新增的元组来反映新增时空单元。
 - ◇ 时空复合模型的基础是时空单元, 即相同时空变化过程的最大单元
 - ◇ 时空过程每变化一次, 即在关系表中新增一列时间段来表达, 实现了用**静态的属性表**来表达**动态的时空变化过程**。
8. **试对比分析离散网格单元列表模型和基态修正模型在时空数据组织中优劣**:
 - 1) 离散网格: 网格单元及其变化以变长列表形式存储, 各个网格单元列表的一个元素对应于该位置上的一次时空变化; 是基于位置模型, 因此, 对于基于时间的查询, 仍需查询所有位置。
 - 2) 基态修正模型按事先设定的时间间隔采样, 不存储研究区域中每个状态的全部信息, 只存储某个时间的数据状态, 以及相对于基态的变化量, 避免连续快照模型将每张未发生变化部分的快照特征重复进行记录。优点: 数据量大大减小、时态分辨率值与事件发生的时刻可以完全对应, 只记录一个数据基态和相对基态的变化值, 提高了时态分辨率, 减少了数据冗余量。保证了地学对象的完整性; 缺点: 仍难以适应基于时间的空间查询。

9. **三域模型**: 数据用三个分离的表存储, 三个表之间通过主键链接, 这个表被称为域链接表 (Domain Link Table)

a. Semantic Table				b. Time Table		
Seq. ID	Landcover	Management	Address	Time ID	Time	Operator ID
1	Old Growth	USFS	12 Forest Rd.	1	1000	2439
2	Clearcut	A Log Co.	3 Forest Rd.	2	1700	2439
3	Burn	USFS	12 Forest Rd.	3	1800	7473
4	Clear-cut	B Log Co.	45 Pine Ave.	4	1950	1929
				5	1960	1929

h. Space Table			d. Domain Link Table (links among temporal, semantic, and spatial objects)		
Space ID	Area	Parameters	Seq. ID	Time ID	Space ID List
4	A ₁	P ₁	1	1	1
6	A ₂	P ₂	1	2	2
8	A ₃	P ₃	2	2	3
9	A ₄	P ₄	3	2	4
10	A ₅	P ₅	1	3	5
11	A ₆	P ₆	2	3	3, 6
12	A ₇	P ₇	1	4	7, 10
13	A ₈	P ₈	4	4	8, 9
14	A ₉	P ₉	1	5	10, 11, 13
15	A ₁₀	P ₁₀	2	5	6, 12
			3	5	4, 14, 15

a. Elements stored in the spatial domain





第五章 · 空间数据存储与空间索引

➤ 物理数据模型以及存储设施

1. 数据库设计的三个层次：

- 1) 概念模型：实体、(多值)属性、关系——高度抽象描述
- 2) 逻辑模型：关系、原子属性、主键和外键——具体实现的描述
- 3) 物理模型：二级存储、文件结构、索引——使用基本组件实现

2. 物理存储介质层次：

1) 基本存储：寄存器、高速缓冲存储器、主存储器

- a) 寄存器 (register)：CPU 的一部分，用于暂存运算中间结果，与运算部件直接连接，速度最快，数量极少。
- b) 高速缓冲存储器 (cache memory)：CPU 的一部分，用于缓存主存储器，操作系统底层管理。
- c) 主存储器 (main memory)：通过总线与 CPU 相连，存储运算所需的数据和指令；随机访问：访问任何存储单元时间相同；易失性：断电丢失。

2) 联机存储 (Secondary storage)：快闪存储器、磁盘存储器

- a) 快闪存储器 (flash memory)：通过外设接口与总线相连，存储永久保留的数据；速度受到存储介质和接口限制；随机访问，非易失性，断电不丢失；文件系统管理，可以通过操作系统在线访问。
- b) 磁盘存储器 (disk memory)：同上，但是机械装置，速度更慢。

3) 脱机存储：光盘存储器、磁带存储器

- a) 光盘存储器 (CDROM/CDR/CDRW/DVD)：脱机存储，保存备份或者历史档案；机械装置，随机访问，速度更低；有标准数据记录格式，操作系统提供文件系统接口访问。
- b) 磁带存储器 (tape)：脱机存储，保存备份或者历史档案；电磁记录原理；机械装置，顺序访问；速度最低，容量价格比最高 (至几百 G)

◇ 速度从上往下越来越慢，但容量越来越大。

3. 二级存储硬件——磁盘驱动器：

磁盘的结构：磁盘 (圆形磁盘片) → 磁道 → 扇区 → 磁盘块



- 1) **磁道 (track)**: 圆形磁盘片上向边缘延伸的带有间距的同心圆环
- 2) **扇区 (sector)**: 每一个磁道中被分成若干等分的区域, 扇区大小由驱动器的厂商设定
- 3) **磁盘块 (页面)**: 是磁盘与主存之间读写数据的最小传输单元
【一个磁盘块有整数倍个扇区, 这个倍数可在磁盘初始化时设定; 磁盘块 (或简称为页面) 是磁盘与主存之间的最小传输单元】
- 4) **柱面 (cylinder)**: 是抽象出来的一个逻辑概念, 简单来说就是处于同一个垂直区域的磁道称为柱面, 即各盘面上面相同位置磁道的集合。

【磁盘读写数据是按柱面进行的, 磁头读写数据时首先在同一柱面内从 0 磁头开始进行操作, 依次向下在同一柱面的不同盘面(即磁头上)进行操作, 只有在同一柱面所有的磁头全部读写完毕后磁头才转移到下一柱面。】

磁盘驱动器: 每一个盘面有一个磁头 (Disk heads) 用来读写, 磁盘主轴高速旋转, 所有的磁头沿着一个方向转动。

磁盘访问代价的影响因素:

寻道时间 (t_s): Move head assembly to relevant track

延迟时间 (t_l): Wait for spindle to rotate relevant sector under disk head

传输时间 (t_t): 读写扇区所需要的时间

三者的比较: $t_s > t_l > t_t$, 传输时间(t_t)通常在磁盘初始化的时候已经固定, 但数据在磁盘上的不同放置策略可以在很大程度上减少寻道时间 (t_s) 和延迟时间 (t_l)。

4. **缓冲区管理器**: DBMS 的一个软件模块, 负责管理主存与二级存储之间的数据传输; 缓冲区管理器需要为高效的事务处理实施一个协议, 确保事务不会因为一部分数据不在主存中而停顿下来。
5. **域、记录 and 文件**: 磁盘上的数据是文件记录 and 域的集合
 - 1) **记录**: 通常小于一个扇区; 一个扇区上会有很多个记录
 - 2) **文件**: 记录的集合, 一个文件可能跨越多个扇区
 - ◇ 一个页面是槽 (slot) 的集合, 每个槽包含一条记录
 - 3) **文件系统**: 文件的集合, 组织成目录

➤ 文件的结构

1. 常见的文件操作：

- 1) 查找 find: key value——与记录匹配的关键值
- 2) 查找下一个 find next: 返回下一条满足条件的记录
- 3) 插入 Insert: 增加一条新记录, 但不改变文件结构
- 4) 找空间数据库中最邻近 Nearest neighbor 的对象

2. 什么是文件结构: 文件组织其记录的方法。【也就是文件如何组织它的数据】

3. 常用的文件结构: 堆 Heap (无序 unordered 无结构 unstructured 文件)、散列 Hashed 文件、有序 Ordered 文件、聚类 Clustered 文件、描述 Descriptions follow 文件

4. 文件结构——堆 (Heap):

- 1) 记录没有特定的顺序;

【无序文件的优点是在进行插入操作时可以很容易地在文件末尾插入一条新记录】

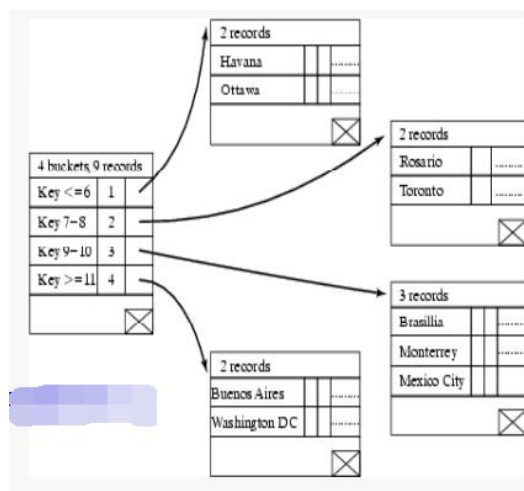
- 2) 文件操作: ①插入: 把记录插入扇区的最后; ②查找、查找下一个、寻找最近邻对象: 扫描整个文件。【平均来说, 需要检索一半的磁盘页面】

5. 文件结构——散列文件 (Hash):

散列函数: 散列函数将事先选择一个主码域 (如 name) 的值映射到一个散列单元中;

采用很简单的计算。【它能够把数量大致相同的记录放入每个散列单元中】

✧ 散列文件组织方式并不适合范围查询, 例如, 查找所有名字以字母“B”开头的城市的详细信息。



6. 文件结构——顺序文件 (Ordered): 有序文件根据给定的主码域对记录进行组织; 查询、插入、删除可以使用 binary search (折半算法) 【查找最方便+最快的方式】

7. 文件结构——聚类文件 (Clustered):

- 1) 有序文件组织方式不能直接应用在空间领域, 有序文件组织方式还可以根据对空间数据集的文件组织方式而概括成空间聚类。【一种替代的方法是使用空间填充曲线面对扇区记录进行聚类】对空间数据库, 意味着在二级存储中, 空间上相邻的和有关联的对象在物理上应当存储在一起。
- 2) 聚类的目的就是降低常见大查询的寻道时间(t_s)和等待时间(t_l)。

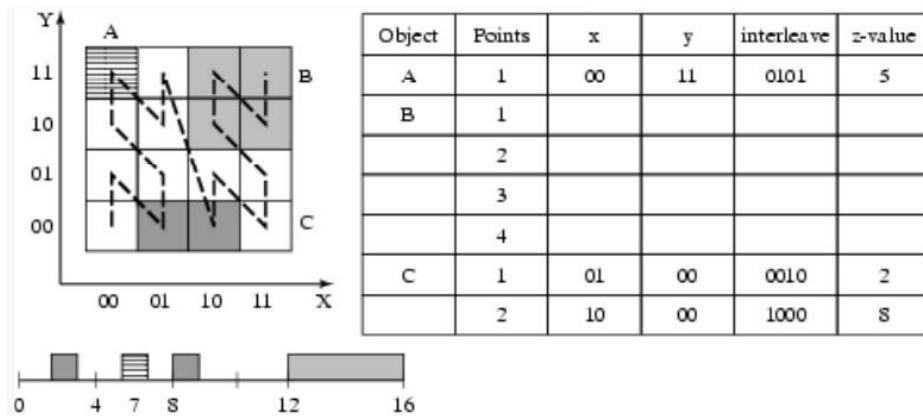
8. 空间数据库管理系统 (SDBMS) 支持的三种聚类方式:

内部聚类 (internal clustering)、本地聚类(local clustering)、全局聚类(global clustering)

9. 空间聚类技术 (又称空间填充曲线): 利用一个线性顺序来填充空间, 可以获得从一端到另一端的曲线。常用的空间聚类技术有: Z 曲线和 Hilbert 曲线。

10. 空间聚类技术——Z 曲线 (Z-curve) 的生成算法:

- 1) 读入 x, y 坐标的二进制表示。
- 2) 间隔扫描二进制数字的比特得到一个字符串。
(先读 x 的第一个再读 y 的第一个再读 x 的第二个.....以此类推)
- 3) 计算结果二进制的十进制值。



☆ 使用 Z 序方法可以处理列出的所有查询:

点查询: 使用二分法在排序文件中查找给出的 z 值;

范围查询: 查询形状可以翻译成一组 z 值, 就像它是一个数据区域一样。通常, 选择它的近似表示, 尽量平衡 z 值的数量和近似中出现额外区域的数量。然后搜索数据区域的 z 值以匹配 z 值。

11. 空间聚类技术——Hilbert 曲线的生成算法:

- 1) 读入 x 和 y 坐标的二进制表示;
- 2) 间隔扫描二进制比特到一个字符串;

		x			
		00	01	10	11
y	00	0000	0010	1001	1010
	01	0001	0011	1001	1011
	10	0100	0110	1100	1110
	11	0101	0111	1101	1111

- 3) 将字符串自左至右分成 2bit 长的串，规定每个 2bit 长的串的十进制值为：“00”等于 0；“01”等于 1；“10”等于 3；“11”等于 2。

		x			
		00	01	10	11
y	00	00	03	30	33
	01	01	02	31	32
	10	10	13	20	23
	11	11	12	21	22

- 4) 对于数组中每个数字 j，如果出现下列两种情况：

- $j=0$ 把后面数组中出现的所有 1 变成 3，并把所有出现的 3 变成 1。
- $j=3$ 把后面数组中出现的所有 0 变成 2，并把所有出现的 2 变成 0。

		x			
		00	01	10	11
y	00	00	01	32	33
	01	03	02	31	30
	10	10	13	20	23
	11	11	12	21	22

- 5) 将上一步得到的数转换成的每一个数字分别用二进制表示，自左至右连接，并计算其十进制值（例如 $32 \rightarrow 1110 = 14$ ）。

		x			
		00	01	10	11
y	00	0	1	14	15
	01	3	2	13	12
	10	4	7	8	11
	11	5	6	9	10

12. **Z 曲线和 Hilbert 曲线的聚类效率比较：**Hilbert 曲线的方法要比 Z 曲线好一些，因为它没有斜线而且二维上相邻的两个点在一维上还是相邻的。不过，Hilbert 算法和精确入口点及出口点的计算量都要比 Z 曲线复杂。

➤ 索引

1. **索引的概念**：索引文件是用来提高数据文件查询效率的辅助文件。

【数据库 = 主文件（数据文件）+ 索引文件】

2. **索引的特性**：

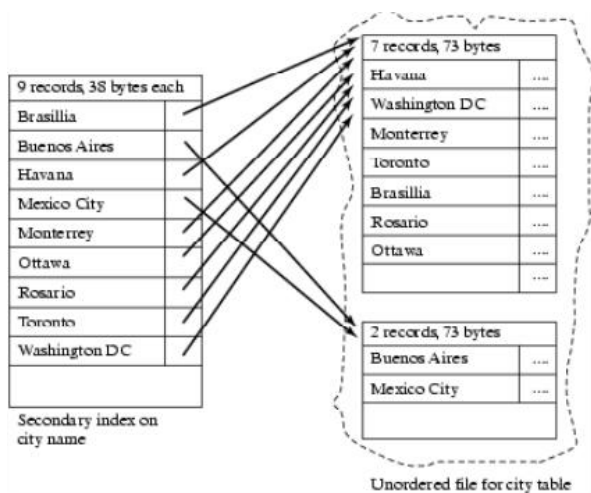
①加速数据库中特定部分访问的数据结构元素；②索引本身也是一种数据结构。

3. **索引表/索引文件**：由索引项构成的表或存储索引项的表称为索引表，以文件方式存储则称为索引文件。【索引表的基本构件是索引项】

4. **索引项**：一个索引项是由关键词值和指针构成的二元组：[关键词值，指针]

1) **关键词值**：记录各种字符段的一个集合，是一个或多个字符段的任意序列组合

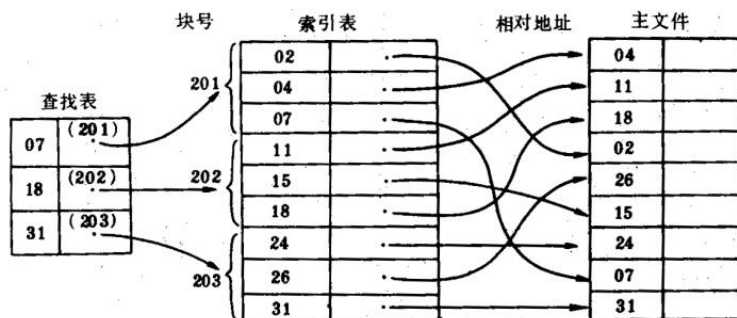
2) **指针**：指向关键词值所标识的记录在数据库中的位置



5. **索引记录是有序记录**：索引文件根据索引属性的值排序，但数据文件本身可以是不按**关键词排序**。【由于索引文件中的记录是有序的，那么即使数据文件无序，也可使用折半查找来搜索索引文件——加快检索速度】

6. **索引层次结构（多级索引结构）**：索引本身也是一个文件，当索引很大时，可将其分块，建立高一层的索引。如此继续下去，直到最高级索引不超过一个块为止。

【索引通常置于磁盘或内存，内存中一般只存放最高级索引】



- ◇ 主索引：如果数据文件的记录是按主码排序的，那么索引就只需要保存数据文件的每个磁盘页面第一个主码域值。



7. 顺序文件和非顺序文件的索引：

1) 索引非顺序文件：

- a) 索引表中顺序列出所有可能的键值（稠密索引），利用二分查找法查找所需键值，得到所需记录地址；
- b) 建立方法：记录按输入的顺序放入数据区，同时软件在索引区建立索引表，待全部数据输完后，软件自动将索引表排序。

2) 索引顺序文件：

- a) 一种按照逻辑键值排序的索引文件，用嵌入索引的手段把顺序文件予以扩充，以加速查找，记录的物理顺序与索引中键值的顺序是一致的。（采用稀疏索引）
- b) 建立方法：数据按顺序分块存放（块间相邻），记录每块的最后记录键值及块的首地址形成索引表。

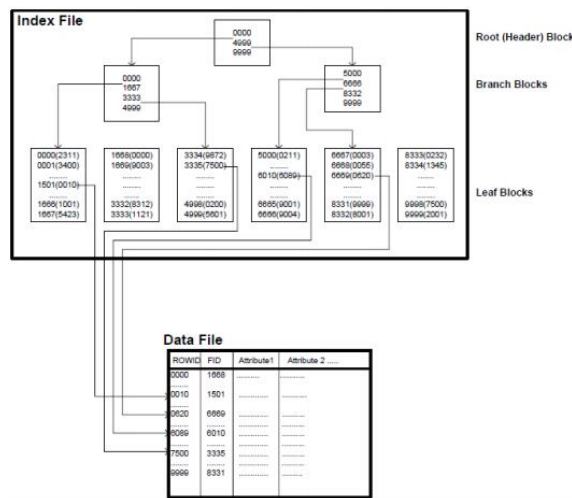
8. 主要的索引技术——B 树索引：B 树索引结构是一种平衡树的多级索引结构，其中的 B 代表着“平衡”（balance，各个分支的深度相同）。

B 树索引的性质：

- 1) B 树索引是由一层或多层的分支节点和一层叶节点组成。
- 2) 分支节点包含了该分支下一层分支的范围信息。
- 3) 叶节点包含实际的索引值和相关行的行标识。
- 4) 根节点和叶节点之间的层数称为索引的深度。

- ◇ B-树索引结构并不含有很多分支节点层，因此，它需要较少的 I/O 操作就可以迅

速查到叶节点。由于所有的叶节点在索引中的深度相同，因而对表中行的检索所需的 I/O 操作数相同，因此在查询方法中，B-树索引的性能相对优越些。



➤ 空间索引

1. **空间索引（空间数据库索引）的概念：**依据空间对象的**位置和形状**或空间对象之间的某种**空间关系**按一定的顺序排列的一种**数据结构**，其中包含**空间对象的概要信息**，如对象的标识、外接矩形及指向空间对象实体的指针。【空间索引描述存储在介质上的数据的位置信息，建立逻辑记录与物理记录间的对应关系】
2. **空间索引的作用：**辅助性的空间数据结构，**介于空间操作算法和空间对象之间**，通过筛选作用，大量与特定空间操作无关的空间对象被排除，从而**提高空间操作的速度和效率**。
3. **空间索引的性能**的优劣直接影响空间数据库和地理信息系统的整体性能，它是空间数据库系统的一项关键技术。
4. **空间索引的基本原理：分割原理**

空间索引的基本原理相似，即采用分割原理，把查询空间划分为若干区域（通常为矩形或多边形），这些区域或单元包含空间资料并可唯一标识。

空间索引的两种分割方法：

- 1) **规则分割：**将地理空间按照规则或半规则方式分割，分割单元间与地理对象相关，地理要素的几何部分可能被分割到几个相邻的单元中——地理对象的描述保持完整，而空间索引单元只存储对象的位置参考信息。
- 2) **基于对象的分割：**索引空间的分割直接由地理对象来确定，索引单元包括地理对象的最小外接矩形（MBR）。

5. 常用空间索引：自顶向下、逐级地划分地理空间，从而形成各种树状空间索引结构。

1) 规则分割方法：规则格网索引方法(Jones, 1997 年)、BSP 树(Fuchs, 1983 年) 和 KDB 树(Robinson, 1987 年)等。

2) 基于对象的分割方法：R 树(Guttman, 1984 年)、R+树(Sellis, 1987 年)和 Cell 树(Guttman, 1991 年；刘东, 1996 年；陈述彭, 1999 年)等。

6. 空间索引——格网索引 (Grid Index)：

基本思想：工作区按一定规则分成大小相等或不等的格网，记录每一个格网所包含的空间对象。【空间格网按 Morton 码(Peano 键)编码，建立 Peano 键与空间对象的关系】

规则格网方法：将嵌入的 n 维空间划分成同等大小的格网。

格网索引的建立步骤：

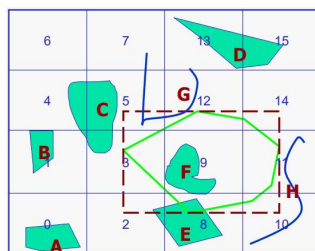
- 1) 将地理空间按照划分行列 ($m \times n$)；
- 2) 计算网格大小及每个格网的矩形范围；
- 3) 开辟目标空间（记录目标穿过的网格）和格网空间（记录格网内的目标）；
- 4) 注册点、线、面、注记等目标，并记录（每个单元格有一个索引记录，登记所有位于或穿过该单元格的物体的关键字）；

■ 索引文件：

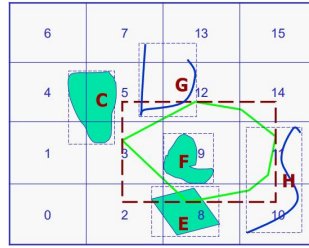
Row ID	Object ID	左上角	右下角
0000	A			
0001	B,C			
0002	E			
0003	C			
0004	C			
0005	C,G			
0006				
0007	G			
.....				

基于规则格网空间索引检索和查询的步骤

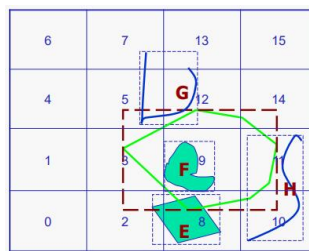
- 1) 根据所接收的 SQL 语句，获取空间过滤器的封装边界所跨越的单元格，然后到空间索引表中检索出封装边界所在单元格内的要素；



- 2) 几何过滤器的**封装边界**与第一阶段检索出的要素的**封装边界**比较，找出具有**重叠**关系的要素；



- 3) 几何过滤器的**坐标**（绿线）与第二阶段检索出的要素的**封装边界**比较，找出封装边界在集合过滤器内的要素；



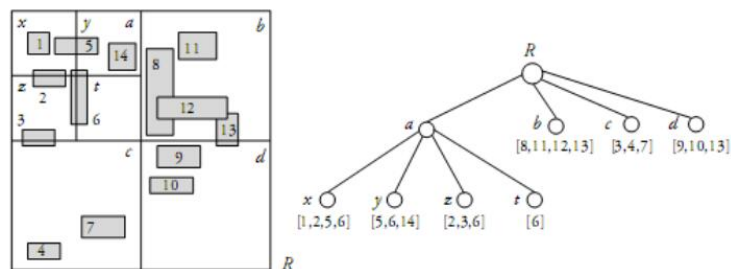
- 4) 几何过滤器的**坐标**与第三阶段检索出的要素的**坐标**（蓝线或绿框）比较，找出最终在几何过滤器内的要素类，并通过索引定位要素在数据库中的位置。

7. 空间索引——四叉树空间索引：

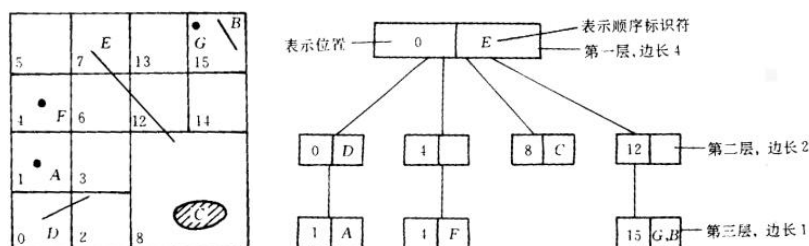
索引思想：递归地对地理空间进行四分，直到自行设定的终止条件（比如每个节点关联图元的个数不超过3个，超过3个，就再四分），最终形成一颗有层次的四叉树。

线性四叉树空间索引分为**层次四叉树空间索引**和**线性四叉树空间索引**两种。

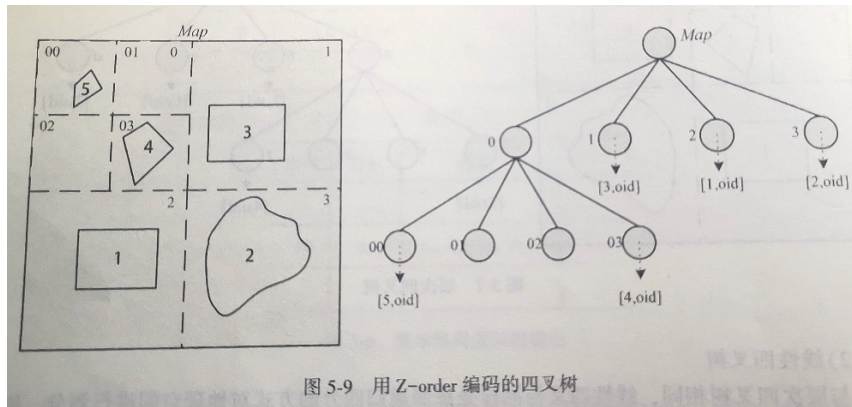
层次四叉树：



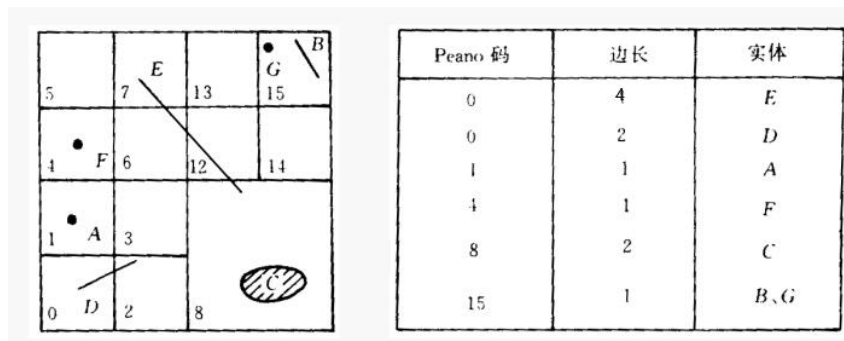
层次四叉树空间索引：中间结点也需要表示出来，而线性四叉树不需要表示中间结点



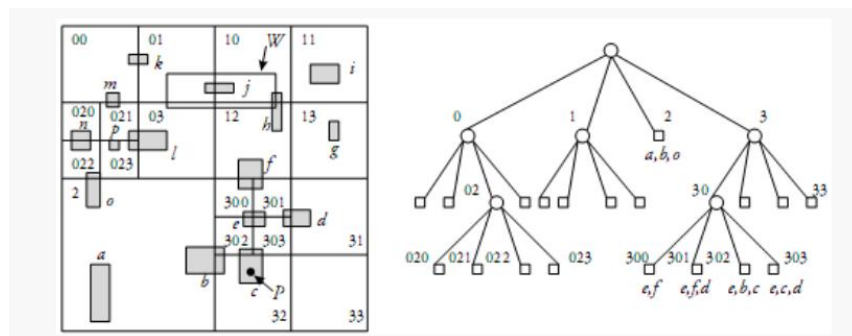
线性四叉树:



线性四叉树空间索引: 根据空间对象覆盖的范围, 进行四叉树分割, 使每个子块中包含单个实体 (或其它终止条件)。根据包含每个实体的子块层数或子块大小, 建立相应的索引。【以第一行为例, 索引表的意思为: 以 0 开始边长为 4 的空间有一个 E】



线性四叉树常用的编码方式: Z 序曲线或 Hilbert 曲线

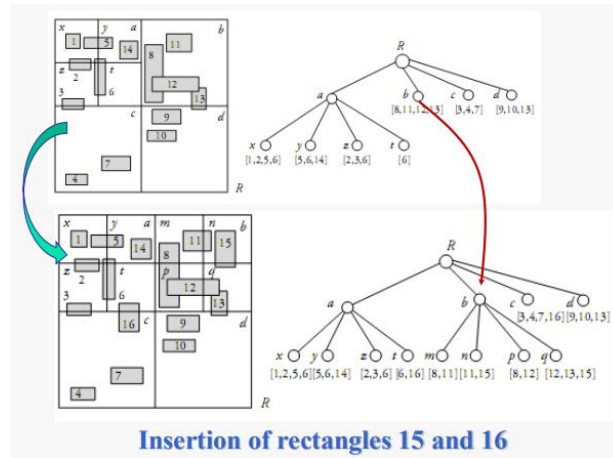


线性四叉树索引建立的注意事项:

- 1) 在四叉树索引中, 大区域空间实体更靠近树的根部, 小实体位于叶端, 以不同的分辨率来描述不同实体的可检索性。
- 2) 为了减少编码的时间, 在数据压缩和 GIS 数据结构中通常采用 **自下而上生成的线性四叉树**。
- 3) **线性四叉树只需存贮最后叶结点信息, 包括叶结点的位置、大小和栅格的属性值。**



四叉树的插入操作：当节点中有空间对象插入时有可能改变索引创建时的终止条件，那么这时该节点就要再次四分。



8. 空间索引——R 树空间索引【基于对象的空间方法】:

- 1) R 树是 B 树在多维空间上的拓展所形成的一种 **多级平衡树**。
- 2) R 树中存放的数据不是原始数据，是这些数据的 **最小边界矩形 (MBR)**，MBR 包含在 R 树的叶结点中。
- 3) R 树空间索引：设计一些 **虚拟的矩形目录**，将空间对象包含在矩形内。以虚拟矩形为空间索引，它包含所含对象的指针。

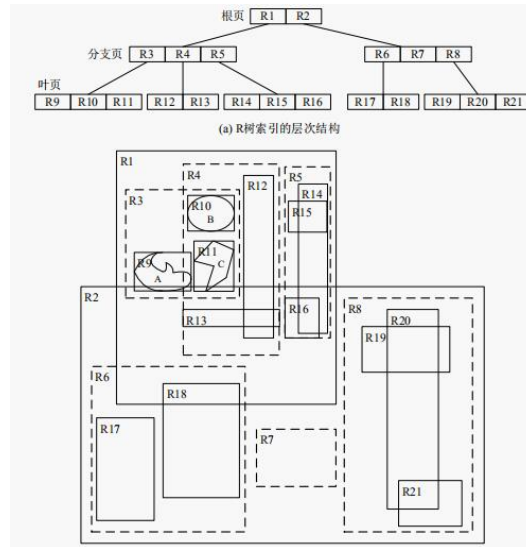
【虚拟矩形的数据结构： $RECT(Rectangle - ID, MinX, MaxX, MinY, MaxY)$ 】

叶结点、中间结点的表示形式：

- 1) **叶结点：**包含索引对象的最小外接矩形和索引标识符，形式为 $(MBR, LeafID)$ 。
 - a) LeafID: 索引对象的唯一标识符
 - b) MBR: 包含该索引对象的最小 n 维外接矩形
- 2) **非叶结点：**入口形式 $(MBR, ChildID)$
 - a) ChildID: 指向下一层子结点的指针
 - b) MBR: 包含下一层所有子结点的最小外接矩形

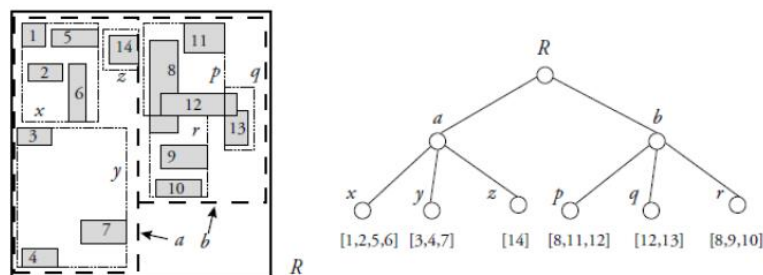
R 树的查询与插入：

- 1) **查询点 (或区域)：**与根结点中每个项进行比较，找到包含该点或与区域交叠的节点，递归地在子结点指针指向的 R 树结点上查找。过程直至 R 树的叶结点为止。用叶结点中选出的项来检索记录。
- 2) 由于 R 树是一棵平衡树，在 **插入对象时**，可能会导致结点向根部分裂。



9. 空间索引——R+树:

兄弟节点对应的空间区域没有重叠，允许一个空间目标被多个虚拟矩形包含



【注意看 12 号，同时在 p 和 q 两个目录矩形下】

特点：空间索引搜索的效率提高，插入和删除效率低。

10. R 树家族的对比:

1) 思路一致：使用 **层次矩形结构** 组织空间数据

2) 二者区别:

a) **R 树索引**：兄弟节点对应的空间区域 **可以重叠**；

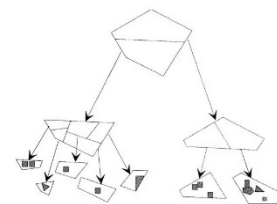
【易进行插入和删除操作；重叠使空间搜索的效率低】

b) **R+树索引**：兄弟节点对应的空间区域 **没有重叠**，**允许一个空间目标被多个虚拟矩形包含**。

11. Cell 树索引：划分空间时采用凸多边形作为划分基本单位，子空间不相互覆盖

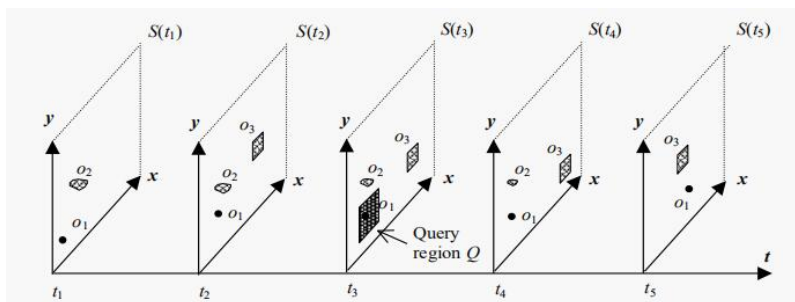
1) Cell 树的磁盘访问次数比 R 树和 R+树少，性能好

2) Cell 树是比较优秀的空间索引方法



➤ 时空索引

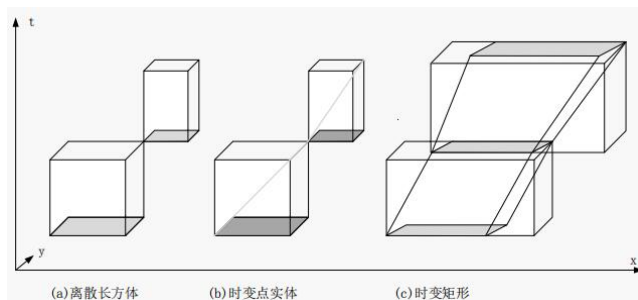
1. **时空数据库的查询**不仅需要关注空间域的问题，还要考虑时间域的问题。时空查询可包括对象“**历史**”查询和“**未来**”查询，都需要从时空数据库中快速查找所要求对象的历史状态。
 - 1) “**历史**”查询：指从时空数据库中查找空间对象的历史演化状态；
 - 2) “**未来**”查询：是查找或预测运动对象的未来位置或状态；其方法主要是根据时空数据库中已经存在的对象的历史状态数据（实数据）在一定的规则和推理模型来进行预测。
2. **时空数据索引创建的基本过程**：将对象的时间和空间范围划分为一些可管理的子空间，子空间进一步被划分为更小的子空间的递归过程。



3. 时空索引——3D 格网索引：

基本原理：将查询地理空间划分为**规则或半规则的单元格**——最简单的时空索引构建方式。时空索引中采用**离散长方体**对时空范围**递归划分**以形成三维格网。

3D 格网索引建立的一般过程：①将时空范围在 x 、 y 和 t 三个方向上用行列线条划分，得到离散长方体($m \times n \times l$)构成的单元格；②计算长方体单元格大小及每个单元格的长方体范围；③开辟目标空间（记录目标穿过的三维格网）和三维格网空间（记录三维格网内的目标）；④注册点、线、面、注记等目标，并记录之。即，每一个三维网格在栅格索引中有一个索引条目(记录)，在这个记录中登记所有位于或穿过该三维网格的物体的关键字。

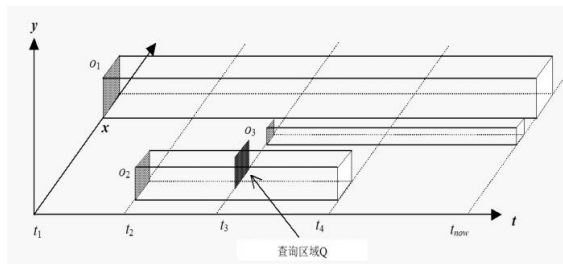


4. 时空索引——3D R 树时空索引：

基本原理：将时间看作另外一维，对 R 树进行扩展而形成的基于对象的时空索引。与 R 树空间索引相对应，3D R 树时空索引使用 3D MBR 来表示单个对象的时空范围。在 3D R 树节点中并不存放原始时空对象数据，而是存储每个节点的 3D MBR。

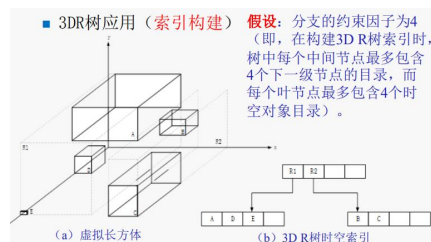
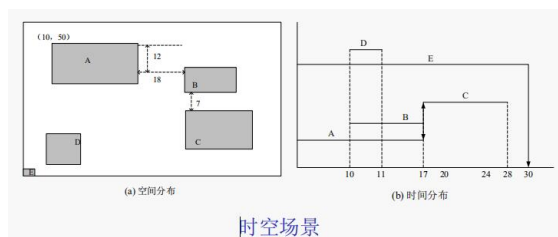
虚拟目录长方体的数据结构可定义为：

$$RECT(Rectangle - ID, MinX, MaxX, MinY, MaxY, MinT, MaxT)$$

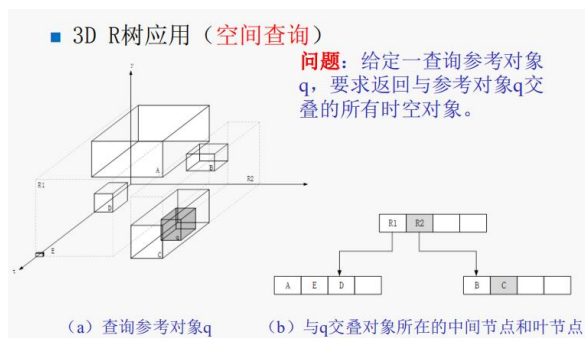


树的应用实例：

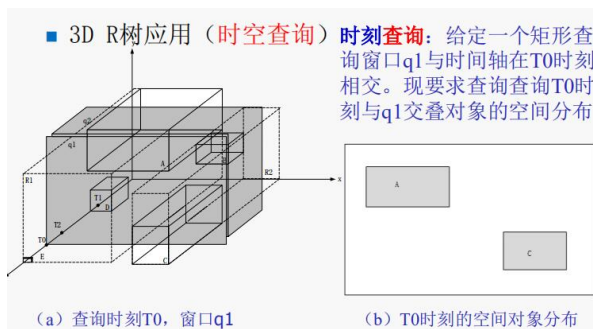
待描述的时空场景以及索引构建：



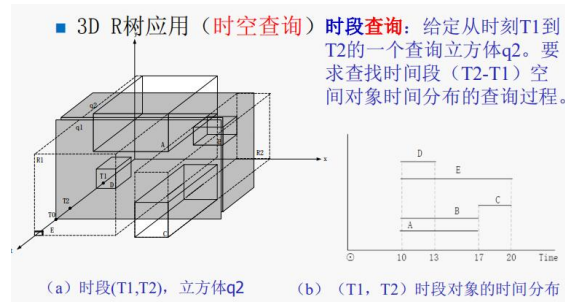
空间查询：



时空查询-时刻：



3) 时空查询-时段:

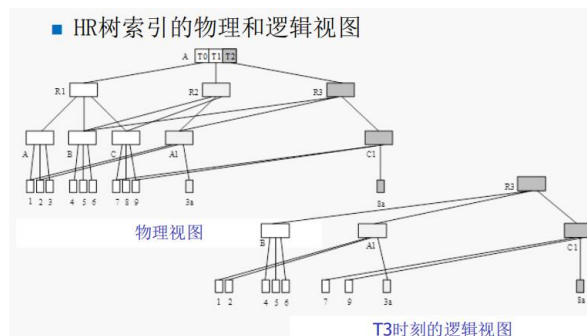


时空索引——HR 树时空索引 (H 为历史 History):

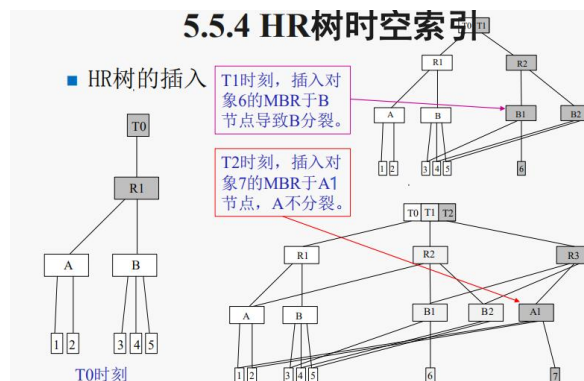
基本原理: HR 树时空索引结构可看作是由时间轴上的一组 R 树逻辑版本视图构成的序列。HR 树索引是由空间维+时间维构成三维索引结构。其中, 空间维是由逻辑 R 树索引来表示, 而时间维则是由时间戳 t 的序列来构成。这里, 一个时间戳就表示一个时间节点。

✧ 为了节约存储空间, 在 HR 树中, 连续时间戳的 R 树空间索引可以共享分支。

HR 树索引的物理和逻辑视图:



树的插入:



树的特点:

- 1) **HR 树的优点:** 对时间戳查询的回答非常有效, 这主要是因为根据 HR 树的本质, 时间戳查询可退化为查询时间窗口对应的 R 树处理的空间窗口查询。

- 2) **HR 树的不足**: HR 树索引存在存储空间消耗太大的问题。因为,即使只有一个对象移动,也需要复制一个节点。HR 树索引在支持区间查询处理的能力上表现出不足。虽然 HR 树通过冗余(节点复制)保持了时间戳查询的良好性能,但它并不能有效支持区间查询处理。

第六章 · 空间查询与访问

➤ 查询与查询语言

1. **查询的定义**: 用户向数据库提出的一个问题或任务。
2. **查询语言**: 表达与数据有关问题的语言。【自然语言、计算机编程语言、结构化查询语言(SQL, 关系和对象关系数据库的标准查询语言)】
3. **SQL 的组成**:
 - 1) **数据定义语言(Data Definition Language, DDL)**: ①数据定义, 数据库和数据库中表的定义, 包括创建、删除和更新; ②索引定义, 索引的创建和删除。
 - 2) **数据操作语言(Data Manipulation Language, DML)**: ①数据库查询, 允许用户检索存储在数据库中的数据; ②数据操作, 允许用户通过插入新的数据、删除历史数据和修改现有的数据值来调整一个数据库的内容。
 - 3) **数据控制语言(Data Control Language, DCL)**
4. 关系代数是与关系模型相关联的形式化查询语言, **数据库中常用的基本运算**: 选择 SELECT、投影 PROJECT、笛卡尔积 PRODUCT、连接 JOIN、并 UNION、差 DIFFERENCE、交 INTERSECT

运 算 符		含 义	运 算 符		含 义
集 合 运 算 符	U	并 差 交	比 较 运 算 符	>	大 于
	-			≥	大 于 等 于
	∩			<	小 于
				≤	小 于 等 于
				=	等 于
				≠	不 等 于
专 门 的 关 系 运 算 符	×	广 义 笛 卡 尔 积 选 择 投 影 连 接 除	逻 辑 运 算 符	¬	非
	σ			∧	与
	π			∨	或
	⋈				
	÷				

➤ 空间查询与空间 SQL

1. **空间查询**：利用一个或多个空间操作算子构成，包括表达空间关系的谓词。

空间查询的分类：

- 1) **几何参数查询**：点的位置坐标，两点间的距离，一个或一段线目标的长度。
 - 2) **空间定位查询**：给定一个点或一个几何图形，检索该图形范围内的空间对象及其属性。
 - 3) **空间关系查询**：空间拓扑关系查询（邻接查询、穿越查询、包含关系查询、落入查询）+ 缓冲区查询。
2. **标准 SQL 时空查询扩展**：Egenhofer 的扩展、OGIS 的扩展、SQL 的时态扩展。
 3. **Egenhofer 的拓展的原则**：①是标准 SQL 的最小扩展，保持 SELECT-FROM-WHERE 结构作为数据库查询的框架；②空间对象的高级处理，能够定义复杂的抽象空间数据类型和它的不同维数的子类型；③空间操作和关系的合并，利用标准 SQL 和空间 SQL 分别查询非空间和空间数据，并指示表示组件控制和显示结果；④通过解释查询结果显示上下文，然后选择和显示与此查询相关的背景资料。

```
SELECT parcel.name  
FROM parcel, subdivision  
WHERE within (parcel.loc, subdivision.loc)  
AND subdivision.name="cranebrook"
```

标准的 SQL 扩展：OGC 定义的面向对象的几何数据模型允许嵌入各种编程语言，OGC 基于所定义的几何数据模型制定了开放式地理空间数据互操作规范（Open Geodata Interoperability Specification, OGIS），把二维地理空间的抽象数据类型（abstract data type, ADT）整合到 SQL 标准之中，实现了对 SQL 标准的空间操作扩展。

5. **SQL 的时态扩展**：

- 1) **时态数据定义语言**：时态数据定义语言用来定义时空数据库中各种时态数据。
 - 2) **时态数据操作语言**：时态数据操作语言用来提供对时空数据库中的时态数据的各种操作。
 - 3) **时态数据查询语言**：时态数据查询语言用来查询时空数据库中的数据，且提供对时态语义的支持。
6. **空间查询运算符**：在空间数据库内容中，空间数据库查询是利用一个或多个运算符构

成的，包括表达空间关系的谓词。

空间运算符的类型 { 按处理的对象数分：一元算子、二元算子
根据对象间的几何关系分类：拓扑算子、投影算子、度量算子

- 1) 一元算子：获取单个几何的属性信息（如，位置、面积、长度和体积属性）
 - 2) 二元算子：获取两个实体间的关系（如距离，方向，邻接性，连接性，包含性）
 - 3) 拓扑算子：使用拓扑关系（如接触，在…内，相交，重叠和无连接）获得一个特定几何的属性。
 - 4) 投影算子：表示凹/凸几何和其他关系（例如，凸包定义为由最小数量的点形成的边界包含的一组点集）的谓词。
 - 5) 度量算子：表示关于测量或几何之间的距离和方向等关系几何间的谓词。
7. OGC (1999) 空间算子定义的几何算子：（回答了空间 SQL 对普通 SQL 的拓展，增加了 3 类空间操作谓词，可以解决什么问题？☆☆☆）

OGIS 规范中定义的几何操作分成三类：

- 1) 适用于所有几何类型的基本操作，例如，SpatialReference 返回所定义对象几何体采用的基础坐标系；
- 2) 用于空间对象间拓扑关系测试的操作，例如，Overlap 判断两个对象内部是否有一个非空的交集；
- 3) 用于空间对象之间分析计算的空间分析操作，例如，Distance 返回两个空间对象之间的最短距离。

类别	操作符	操作符功能
基本操作符	Spatial Reference Envelope Export IsEmpty IsSimple Boundary	返回几何体的参照系统 返回几何体的最小外包矩形 将几何体转换成其他表示格式 测试几何体是否为空 如果几何体是简单几何体则返回TRUE 返回几何体的边界
拓扑操作符	Equal Disjoint Intersect Touch Cross Within Contain Overlap Relate	判断几何体是否在空间上相等 判断几何体是否相离 判断几何体是否相交 判断几何体之间是否接触 判断几何体是否相互交叉 判断几何体是否包含于另一个几何体之中 判断几何体是否包含了另一个几何体 判断几何体是否覆盖了另一个几何体 如果存在9交（9-Intersection）矩阵描述的空间联系则返回TRUE
空间分析操作符	Distance Buffer ConvexHull Intersection Union Difference SymDifference	返回分别取自两个几何体上的两点的最短距离 返回一个缓冲几何体，它的所有点与原几何体的距离小于或等于某阈值 返回几何体的凸包 返回两个几何体的相交区 返回两个几何体的联合区 返回两个几何体的不同区 返回两个几何体的对称差分（如逻辑运算“异或”）

8. Oracle Spatial 提供不同的操作符来执行下述的邻近分析：

1) 找出在查询位置指定距离内的所有数据：该操作符称为

SDO_WITHIN_DISTANCE 或简称为 within distance 操作符。

✧ 给定一个位置集，SDO_WITHIN_DISTANCE 操作符将从中返回在一个查询位置指定距离范围内的所有位置。

2) 找出距离查询位置最近的邻居：该操作符被称为 SDO_NN 或简称为

nearestneighbor 操作符。

✧ SDO_WITHIN_DISTANCE 操作符不适合获取指定数量的邻居，不管他们离查询位置多远，而 SDO_NN 可以。给定一个位置集，SDO_NN 操作符将按其查询位置的距离顺序来返回数据。

3) 找出与查询位置相交或关联的所有邻居：可达到这个目标的主要操作符被称为

SDO_RELATE。如果只使用索引的近似值，那么可以使用一个简单的变种（操作符），称为 SDO_FILTER。

✧ SDO_FILTER 操作符：可以识别出其最小外包矩形与查询几何体的最小外包矩形有相互作用的所有几何体，主要用于空间索引，不调用 Geometry Engine 函数；

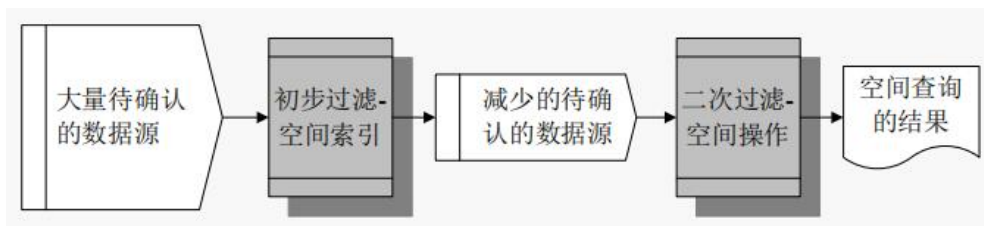
✧ SDO_RELATE 操作符：可识别出以某种指定的方式与查询几何体相互作用的所有几何体（指定的相互作用方式可以是相交、与边界接触、完全包含在内等）。

9. 时间运算符：

利用空间几何对象的时态操作和时态关系提出时空数据库中时态查询运算符。

类别	操作符	运算符功能
基本操作符	Instant	返回指定时刻/时间点的空间对象
	Interval	返回指定时间段内的空间对象
	Nearest Neighbor, NN	在运动对象查询中返回对于给定对象q和对象集 $P=\{p_1, p_2, \dots, p_m\}$ ($m \geq 1$), 求满足 $ q, p_i $ (p_i 属于 P) 最小的 p_i . q和 p_i 都是静止的。
时态拓扑操作符	Equals	判断几何体的状态是否在时间上相等
	Before/After	判断几何体时间的先/后关系
	Meets/Met	判断几何体的状态在时间上是否相交
	Overlaps/Overlapped	判断几何体的状态之间在时间上是否有重叠
	During/Contain	判断一个几何体的状态时间是否有包含
	Starts/Started	判断几何体状态的起点/开始时间是否相同
	Finishes/Finished	判断几何体状态的终点/结束时间是否相同

10. **空间/时态/时空拓扑运算的区别**：**空间拓扑运算**用于分析几何体在空间中的相对位置，**时态拓扑运算**用于分析几何体在时态空间中的相对位置，而**时空拓扑运算**则用于分析在时空对象在时空空间中的位置关系。
11. **空间连接**：当两个表 R 和 S 基于一个空间谓词进行连接时，连接称为空间连接。例如：找到距离河流 50km 以内的所有城市（cities rivers join [dist(center, route) < 50]）
✧ 可用于创建十分强大的空间分析和建模工具，**通常是 GIS 的强制性功能**。
12. **空间查询策略**：一般是将一个或多个过滤（过滤：减少第二阶段密集的计算量）方法用于加速数据库的访问过程。



第一次过滤（过滤步骤）：数据库的空间索引，找到与查询有可能有关的空间对象。

第二次过滤（精化步骤）：一次或多次的空间操作，用 GIS 处理集合找到查询的答案

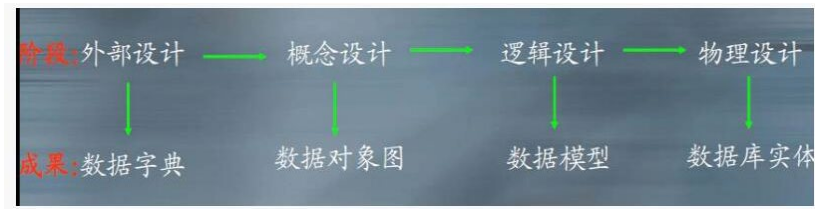
第七章 · 空间数据库设计

➤ 空间数据库设计主要内容

1. **空间数据库设计的主要内容**：确定 SDBMS 支持的最优数据模型、处理模式、存储结构和存取方法，实现对应用系统有效的管理，满足用户信息要求和处理要求。
2. **设计的分类**：
 - 1) **静态设计（结构特性设计）**：设计数据库的数据模型或数据库模式，包括概念结构设计和逻辑结构设计；
 - 2) **动态设计**：数据库行为特性设计，设计数据库查询、事务处理和报表处理
 - 3) **物理设计**：对数据模式的物理实现，即设计数据库的存储模式和存取方法。
3. **设计的要求**：①空间数据库设计与应用系统设计相结合（空间数据库设计与数据的处理设计密切结合）；②数据独立性（逻辑独立性、物理独立）；③合理的数据冗余，提高共享程度；④简单的接口；⑤系统可靠、安全与完整性；⑥有重新组织数据的能力；⑦系统可修改和可扩充性。

➤ 空间数据库设计过程

1. 空间数据库的设计过程：



2. 外部设计阶段：根据实际应用需求将数据对象分析和简化，获取用户的数据要求。

- ✧ 外部设计的关键任务：找出数据中公有的、核心的对象，得到描述数据对象内容的数据字典。
- ✧ 空间数据库的需求分析包括：用户的基本需求、空间数据的现状、系统环境/功能分析。
- ✧ 系统需求分析强调两点：①需求分析阶段一定要**收集将来应用所涉及的数据**；②需求分析必须**要有用户参与**。
- ✧ 需求分析的工具：①**数据流图**（以图形的方法描绘数据在系统中流动和处理的过程）；②**分层数据流图**；③**数据字典**（定义数据流图中的各个成分的具体含义，是数据信息的集合）。

3. 概念设计阶段：构建数据模型表达数据对象的结构特征和相互关系，分析和表达数据对象的空间、时间和属性特征。（ER 图/UML 类图）

- 1) 概念模型的特点/要求：①概念模型是对现实世界的一个真实模型。②概念模型**应当易于理解**。③概念模型应当**易于修改**。④概念模型应易于向数据模型转换。

2) 概念模型中实体之间的基本定义：

- a) 实体列表：是数据实体或其属性按某种易读形式进行编排
- b) E-R 图：实体联系模型（E-R 模型）、用象形图扩展的 E-R 模型

- ✧ **ER 模型表达空间概念的不足之处**：ER 模型的设计是基于对象模型的，场模型无法用 ER 图进行自然的映射。

3) 信息——实体、属性、联系——数据模型

4. 对 ER 图进行扩展：增加某种结构来接受和表达空间推理的语义，同时保持图形表示的简洁性——**象形图（pictogram）注释和扩展 ER 图**：

- 1) **实体象形图**：是一种将对象插在方框内的微缩图表示，可以是基本形状，也可以

是自定义的形状。

- a) **基本形状**: 矢量模型的基本元素有点、线、多边形
- b) **复合形状**: 定义聚合形状, 用基数量化这些复合形状, 0 表示无法在某个给定比例尺下描绘的要素
- c) **导出形状**: 一个对象的形状由其他对象的形状导出, 用斜体形式表示
- d) **备选形状**: 用于表示某种条件下的同一个对象
- e) **任意形状**: 用通配符 (*) 表示, 表示各种形状

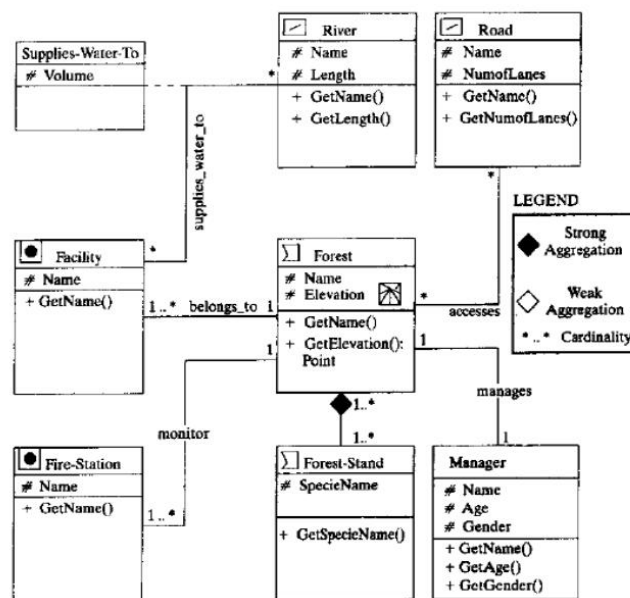
2) **联系象形图**: 用来构建实体间联系的模型



3) **象形图增强了 ER 图对空间语义的表达能力**

5. **UML**: 面向对象的概念模型建模标准, 对结构化模式和动态行为进行建模。

✧ 用类图(class, association, multiplicity)表示概念模型



6. **逻辑设计阶段**: 任务包括: ①完整性约束; ②空间数据的完整性约束; ③用什么数据模型; ④概念模型向逻辑模型的映射。(选择是对象关系模型还是地理关系模型)

7. **完整性约束**:

- 1) **数据库中的完整性**: 指保护数据库中数据的正确性、有效性和一致性, 防止错误的数据库造成无效操作。

- 2) **完整性约束**: 为了保证数据库的完整性, 数据库管理系统制定了加在数据库数据之上的语义约束条件, 即数据库完整性约束条件或完整性管理规则, 这些约束条件作为表定义的一部分储存在数据库中。
- 3) **完整性检查**: DBMS 中检查数据是否满足完整性条件的机制
- 4) **完整性管理规则**: 检查数据库中存放的数据是否满足管理规则, 规则定义了数据库中数据的检查时间、检查内容和违约响应等事项。
 - a) **检查时间**, 指完整性检查的触发条件, 即规定数据库管理系统何时使用完整性约束条件机制对数据进行检查。
 - b) **检查内容**, 指检查的约束条件, 即检查用户的操作请求违背了什么样的完整性约束条件。
 - c) **违约响应**, 指查出错误的处理方式, 如果用户的操作请求违反了完整性约束条件, 应采取一定的措施来保证数据的完整性。
 - d) **完整性规则的表示**: 一条完整性规则, 用一个五元组 (D,O,A,C,P) 来表示。
D (Data), 数据作用的约束对象; O (Operation), 触发完整性检查的操作;
A (Assertion), 数据对象需要满足的语义约束; C (Condition), A 作用的数据对象的谓词; P (Procedure), 违反完整性规则时触发执行的操作过程。
- 5) **完整性控制**: 指数据库管理系统检查用户发出的操作请求是否违背了完整性约束条件, 对违背了完整性约束条件的操作采取一定的动作来保障数据库数据完整性的过程。
- 6) **完整性约束执行时间**:
 - a) **立即执行约束 (Immediate Constraints)**, 指在执行用户事务的过程中, 在某一条语句执行完成时, 系统立即对此数据进行完整性约束条件的检查, 以确保数据的完整性。
 - b) **延迟执行约束 (Deferred Constraints)**, 指在执行完用户的全部事务时, 再对约束条件进行检查, 通过检查的数据才能够提交到数据库中。
8. **空间数据的完整性约束**:

Cockcroft (1997) 扩展了这一类约束以涵盖空间数据的特殊要求。

 - 1) **拓扑完整性约束**, 关于空间要素间空间关系 (如邻接、包含等) 的几何属性。
 - 2) **语义完整性约束**, 控制数据库中对象空间行为的数据库规则 (例如, 地块不能位

于水体中)。

- 3) **用户定义约束**，类似于那些在非空间数据建模中确定的业务规则（例如，沿湖岸200米的缓冲区内禁止树木采伐、或禁止建居住房屋等）。

Cockcroft(1998)进一步提出，上面三类约束性条件中的每一个都可以既应用于一致性状态的数据也可以应用于事务处理中的数据。这样就导致了下面六类空间数据完整性约束：①静态的拓扑完整性约束；②变换拓扑完整性约束；③静态语义完整性约束；④变换语义完整性约束；⑤静态用户定义完整性约束；⑥变换用户定义完整性约束。

9. 完整性约束应用（以 ESRI Geodatabase 为例）：

1) 面规则：

- a) Must not overlap：要求面的内部不重叠
- b) Must not have gaps：此规则要求单一面之中或两个相邻面之间没有空白。所有面必须组成一个连续表面。
- c) Must not overlap with：要求一个要素类（或子类型）面的内部不得与另一个要素类（或子类型）面的内部相重叠

2) 线规则：

- a) Must not overlap：要求线不能与同一要素类（或子类型）中的线重叠。
- b) Must not intersect：要求相同要素类（或子类型）中的线要素不能彼此相交或重叠。
- c) Must not overlap with：要求一个要素类（或子类型）中的线要素不能与另一个要素类（或子类型）中的线要素相交或重叠。

3) 点规则

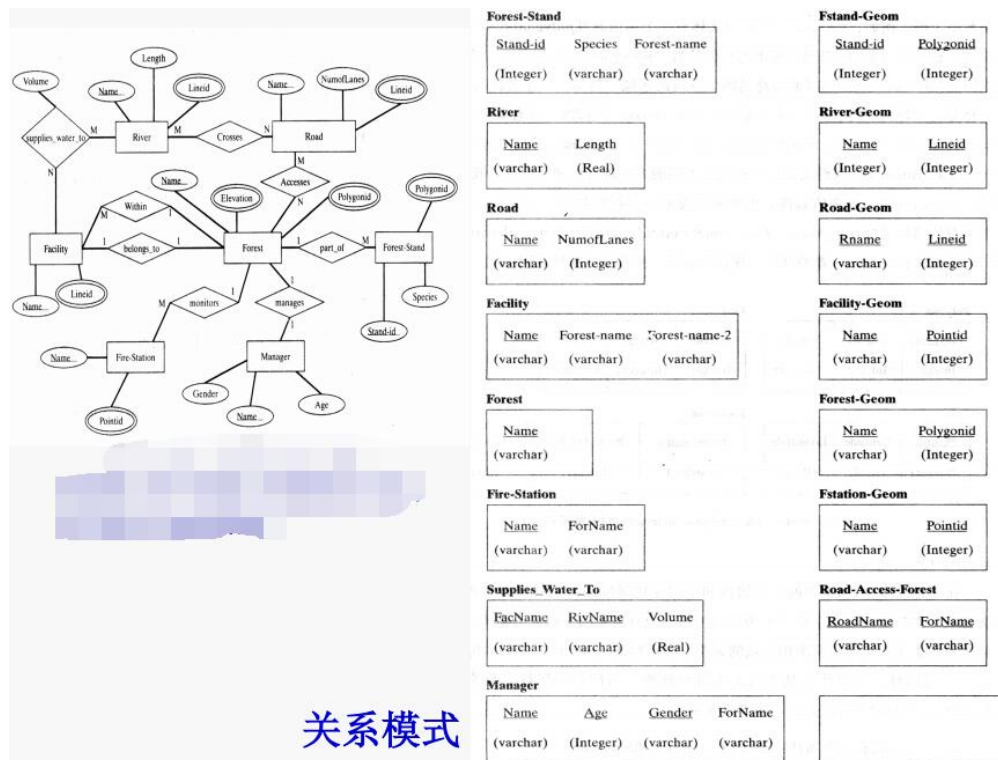
- a) Must coincide with(必须重合)：要求一个要素类（或子类型）中的点必须与另一个要素类（或子类型）中的点重合。
- b) Must be disjoint（必须不交）：要求点与相同要素类（或子类型）中的其他点在空间上相互分离。重叠的任何点都是错误。
- c) Must be covered by boundary of（必须被其他要素的边界覆盖）：要求点位于面要素的边界上。

10. **时态数据库完整性**：为了保证时态数据库中存储时态数据的正确性（符合现实世界的语义）而确立的规则。工作可归纳为4个方面：

数据和对应时间的完整性；时态实体完整性和时态参照完整性；静态完整性（传统数据库的完整性）；动态完整性（时态数据库的完整性）。

11. ER 模型（概念模型）向关系模式（逻辑模型）的映射的转换规则：

- 1) 每一个实体都转换成关系；将 ER 图中的所有信息都变成表。
- 2) 一对一的联系：将任一实体的码属性作为其他关系的一个外码。
- 3) 多对一的联系：将“1”侧的关系的主码作为“M”侧关系的外码。
- 4) 多对多的联系：M 和 N 共同形成新表的关键字。



12. 物理模型设计阶段：解决数据的存放结构和存取方法的问题。（磁盘、文件结构）

13. 系统实施阶段：①建立实际的数据库结构；②装入实验数据对应用程序进行测试；③装入实际数据建立实际数据库。

➤ 事务管理

1. 事务(Transaction)：是用户定义的一个数据库操作序列，是引起数据库值改变的数据库操作。这些操作要么全做，要么全不做，是一个不可分割的工作单位。

【事务是恢复和并发控制的基本单位，一个应用程序通常包含多个事务】

2. 事务与查询的区别：数据库查询返回了数据库中指定数据项的值，并没有改变数据项的初始值。相反的，当数据项的值发生改变时，则，被认为发生了数据库事务。



3. 数据库事务的特性:

- 1) 原子性, 一个事务要么被执行, 要么没有事务发生 (也就是说, 一个事务不能被部分完成);
- 2) 一致性保持 (Consistency preservation), 一个数据在数据库事务发生前后的保持性, 即处于一个由数据库模式和其他的约束或作用于数据库上的完整性规则所规范的“一致性状态 (consistent state)”;
- 3) 孤立性 (Isolation), 要求并发事务的结果是相互独立的。
- 4) 耐久性或永久性 (Durability or permanency), 事务完成后, 其结果可以并始终被追溯, 即使系统失败或崩溃。

4. 长事务冲突的种类:

- 1) 更新冲突, 对于两个并发的事务, 如果它们都更新了同一图层的同一图形元素, 则称这两个事物是更新冲突的。
- 2) 同一冲突, 对于两个并发的事务, 如果它们在同一图层中创建了不同的图形元素而描绘的是同一现实对象, 则称这两个事务是同一冲突的。
- 3) 关系冲突, 应用模型可能对一个或多个图层中图形元素之间的关系有一定的约束, 而违反约束导致的冲突。
- 4) 其他冲突, 除了上述三种冲突外, 并发长事务还可能存在其他类型的冲突, 如读写冲突。

5. 版本: 通过各种状态来明确记录单个要素或对象的版本作为它们被改变、添加和退役的过程。

- 1) **Default 版本。**为根版本, 它是其他所有版本的祖先版本。与其他版本不同, Default 版本始终存在, 且不能被删除。
- 2) **增量表。**版本在被称为“Adds”和“Deletes”的增量表 (delta tables) 清楚记录地理数据库中对象的状态。 增量表 (delta table) 模式用于支持数据集的同步编辑以及大量的基于版本的 GIS 工作流。
- 3) **基表。**基表是要素类的核心表。基表也称为业务表, 它包含所有非空间属性, 如果使用 SQL 几何类型, 则它还包含空间属性
- 4) **子版本。**子版本是通过其他版本创建的地理数据库版本。这里其他版本则为父版本。

6. 冲突原因与解决机制：

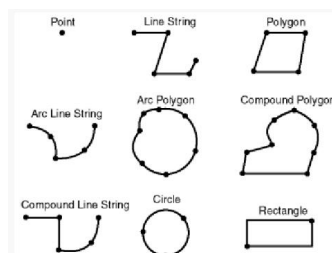
冲突原因：①当前正在编辑的版本和目标版本中对同一要素进行更新；②在一个版本中更新某个要素，同时在另一版本中删除该要素；③在当前正编辑的版本和目标版本中修改拓扑结构上相关的要素或关系类。

解决机制：属性替换、要素替换、类级别替换、完全替换、合并几何。

第八章 · 商用空间数据库

➤ Oracle Spatial

1. **Oracle Spatial** 是一系列函数和过程的集合，提供了 SQL 模式和函数来实现要素集的存储、检索、更新和查询【**是一种对象关系数据库，是对象关系数据模型的实现**】
2. **Oracle Spatial 的组成：**
 - 1) 实现模式 (MDSYS)：规定了支持的几何数据类型的存储、语法和语义
 - 2) 空间索引机制：建立对空间数据查询的空间索引
 - 3) 一套运算符和函数：支持对空间数据的操作
 - 4) 管理工具：
3. **Oracle Spatial 的层次结构数据模型：**高层次的对象由低一层次的对象构成，包括元素 (element)，几何形 (geometry) 和图层 (layer) 三个层次
 - 1) **元素：**构成几何形对象的零件，包括点、折线段和多边形
 - ✧ 多边形按照顶点排列顺序分为外部多边形和内部多边形，顺时针为内，逆时针为外
 - 2) **几何形：**代表一个地物，由若干顺序排列的元素构成，类型可相同也可不同
 - 3) **图层：**若干具有相同属性的几何形的集合
4. **Oracle Spatial 的 9 种几何对象类型：**①点 (集) Point (cluster)；②折线 Line string；③多边形 Polygon；④弧线 Arc line string；⑤弧多边形 Arc polygon；⑥复合多边形 Compound polygon；⑦复合折线 Compound line string；⑧圆 Circle；⑨矩形 Rectangle



5. Oracle Spatial 支持四种坐标系：笛卡尔坐标系、地理坐标、投影坐标、局部坐标。
 - 1) 笛卡尔坐标 **Cartesian coordinate**: 平面直角坐标，是几何形的缺省坐标系
 - 2) 地理坐标 **geodetic coordinate**: 经纬度坐标，与大地基准点相关
 - 3) 投影坐标 **projected coordinate**: 对应不同的应用目的，有多种投影方式
 - 4) 局部坐标 **local coordinate**: 非地理应用，如 CAD 等
6. Oracle Spatial 主要通过元数据表、空间数据字段（即 SDO_GEOMETRY 字段）和空间索引管理空间数据，并提供一系列空间查询和空间分析的函数，让用户进行更深层次的 GIS 应用开发。
7. 空间字段 SDO_GEOMETRY:

```
create type SDO_GEOMETRY as object (
  SDO_GTYPE number,
  SDO_SRID number,
  SDO_POINT SDO_POINT_TYPE,
  SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,
  SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY
);
```

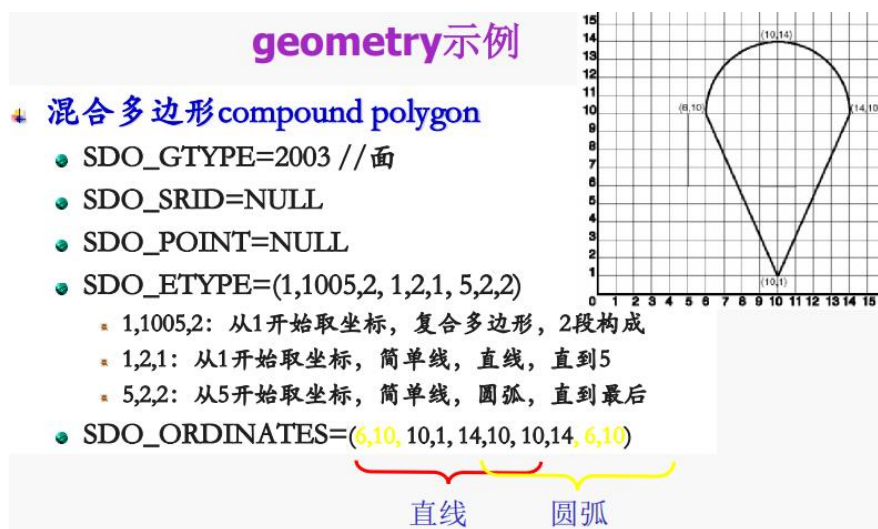
- 1) **SDO_GTYPE**: 指明了 geometry 的类型，对应 OGIS 几何对象模型中的类型。是一个 4 位数字，格式为 dltt:
 d 代表 geometry 的维数:2,3,4; 1 代表线性参照系 LRS 测量维度，非 LRS 为 0;
 tt 指明了 geometry 的类型，00-07, 08-99 保留

值	geometry 类型	说明
d100	UNKNOWN_GEOMETRY	Spatial 忽略本类型
d101	POINT	点
d102	LINE or CURVE	折线或者带圆弧折线
d103	POLYGON	多边形
d104	COLLECTION	异质 geometry 集合
d105	MULTIPOINT	点集
d106	MULTILINE or MULTICURVE	线集
d107	MULTIPOLYGON	多边形集

- 2) **SDO_SRID**: 用来确定 geometry 所对应的空间参照系
 null 代表不对应任何参照系；非空值必需参照 MDSYS.CS_SRS 表的 SRID 字段，同时需要存入 USER_SDO_GEOM_METADATA 表的 SRID 字段；在同一个字段里的所有 geometry 对象都必须具有相同的 SRID
- 3) **SDO_POINT**: 如果 SDO_ELEM_INFO 和 SDO_ORDINATES 成员都为空的话，

SDO_POINT 就是一个 POINT 对象的坐标值，否则 SDO_POINT 被忽略

- 4) **SDO_ELEM_INFO**: 变长的数值数组，说明 SDO_ORDINATES 成员中数值的意义。每三个数值为一个单元，分别是：
 - a) **SDO_STARTING_OFFSET**: 元素坐标值从 SDO_ORDINATES 数组第几个数开始 (SDO_STARTING_OFFSET >= 1)
 - b) **SDO_ETYPE**: 说明元素的类型
1/2/1003/2003 表明是一个简单元素；4/1005/2005 表明是一个复合元素
 - c) **SDO_INTERPRETATION**:
当 ETYPE=1003/2003，指明解释 SDO_ORDINATES 的方式；
当 ETYPE=1005/2005，指明后面的几个单元构成复合元素。
- 5) **SDO_ORDINATES**: 是一个可变长度的数组，用于存储几何对象的真实坐标，该数组的类型为 NUMBER 型，它的最大长度为 1048576。必须 SDO_ELEM_INFO 数组配合使用，才具有实际意义。

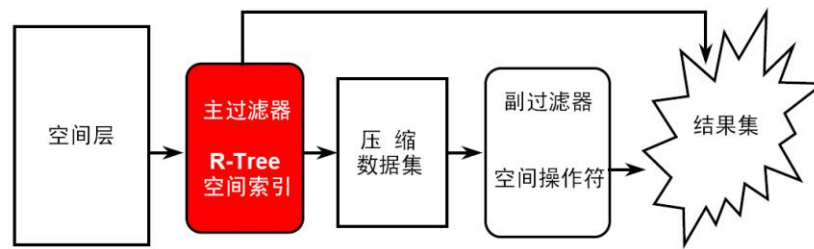


8. **元数据表**: 存储有空间数据的数据表名称、空间字段名称、空间数据的坐标范围、坐标参考信息以及坐标维数说明等信息，一般通过元数据视图 (USER_SDO_GEOM_METADATA) 访问元数据表。

元数据视图的基本定义:

- 1) **TABLE_NAME**: 含有空间数据字段的表名
- 2) **COLUMN_NAME**: 空间数据表中的空间字段名称
- 3) **DIMINFO**: 按照空间维顺序排列的 SDO_DIM_ELEMENT 对象的动态数组
- 4) **SRID**: 用于标识与几何对象相关的空间坐标参考系。

9. Oracle Spatial 空间数据库查询实现的策略：



10. Spatial 采用两级查询模型来实现空间查询和空间连接：空间查询分为两个独立的操作，两次操作的结果合成为最终结果。主过滤 Primary filter 和次过滤 Secondary filter

1) 空间索引实现主过滤，快速从大量数据集中找出查询结果的候选集

Oracle Spatial 的两种空间索引：R 树 R-tree、四叉树 quadtree

R-tree	Quad-tree
几何形的近似形不可调节，采用最小包围盒	可以通过设定分片级别和数量来调节几何形的近似形
索引的创建和调整容易	调整较复杂，会显著影响性能
相对较少的存储空间	相对较大
对最近邻居SDO_NN查询较快	对最近邻居查询较慢
更新数据的效率较低	频繁更新数据不会影响索引的性能
可以多到四维空间索引	只能在二维空间索引
SDO_WITHIN_DISTANCE查询效率高 能够实现全球的索引	

2) Oracle Spatial 的次过滤方法：①SDO_RELATE——测试拓扑关系，采用 9 交叉模型来表达点线面之间的空间关系；②SDO_WITHIN_DISTANCE——测试两个空间对象是否在指定的距离范围内【先创建 B 的缓冲区 D，再判定 A 和 D 是否相离】；③SDO_NN——确定一个空间对象的最近邻居【返回指定个数的距离 A 最近的对象】

11. Oracle 空间聚集函数：空间聚集函数作用于包含若干几何对象的查询结果，返回单个 SDO_GEOMETRY 对象。

空间聚集函数包括了：SDO_AGGR_CENTROID、SDO_AGGR_CONVEXHULL、SDO_AGGR_LRS_CONCAT、SDO_AGGR_MBR、SDO_AGGR_UNION

12. Oracle 地理编码 geocoding：将邮政地址转换为标准化的地址、坐标位置。

13. 空间连接：空间连接查询将两个字段的几何对象作笛卡尔积，两两进行关系判别。可以回答诸如：哪些高速公路穿越了哪些国家公园？这样的问题【空间连接对所用的空间索引要求一致，要么都是 R 树索引，要么都是四叉树索引且层次相同】

➤ 空间数据引擎

1. 实现对地理空间数据管理的方法：

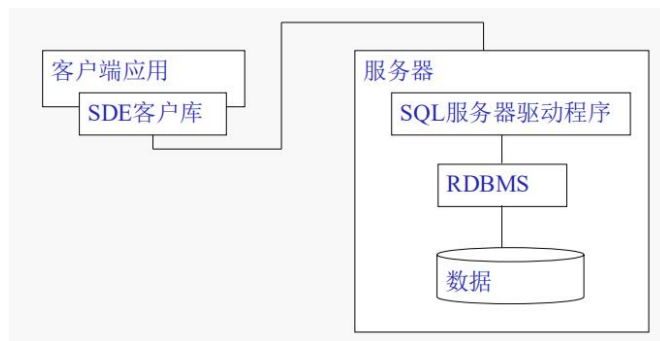
- 1) 从底层开始开发新的数据库管理系统实现对空间数据的管理；
- 2) 以当前通用商用关系型数据库为基础，对其进行扩展使之可以处理复杂的空间数据（关系数据库管理系统+高级语言接口技术）【将复杂的空间实体操作封装成类——地理空间数据库引擎】

2. RDBMS 接口技术解决：数据库管理者或应用程序与数据库的数据交换问题。

3. 空间数据库引擎 SDE：基于特定的空间数据模型，在特定的数据存储、管理系统的基础上，提供对空间数据的存储、检索等操作，以提供在此基础上二次开发的程序功能集合。

4. SDE 的体系结构：

- 1) 相对于客户端，SDE 是服务器，提供空间数据服务的接口，接受所有空间数据服务请求；
- 2) 相对于数据库服务器，SDE 则是客户端，提供数据库访问接口，用于连接数据库和存取空间信息



5. SDE 的发展现状：

- 1) GIS 厂商开发的空间数据管理模块：寄生在关系数据库管理系统之上的空间数据引擎（ESRI : ArcSDE、MapInfo: Spatial Ware）
- 2) 由数据库厂商研发的直接扩展通用数据库的空间数据库系统：Oracle Spatial、Informix Spatial DataBlade、IBM DB2Spatial Extender

6. SDE 的特点：

- 1) 对地理数据的开放式系统访问，使地理数据更易于获得、更易于管理。
- 2) 对用户需求的充分回应。

- 3) 支持大型数据库: SDE 利用统一的数据模型,维护关系数据库中的空间和属性数据,管理近乎无限的空间特征。
 - 4) 进行高效空间查询分析: 可以进行近乎无穷的空间分析,各种空间查询可通过 SQL 的 Where 子句进行。空间查询的结果可以用于制图或其它几何分析应用,即可以把空间分析嵌入到一个非 GIS 的应用程序中去。
 - 5) 理想的空間对象模型
 - 6) 快速实现过程: 对复杂的空間查询来说, SDE 比其它任何空間分析技术完成次要 (subsecond)特征的检索时间要快得多。
 - 7) 网络访问: SDE 支持对 TCP/IP 网络环境的访问。
 - 8) 平台支持: SDE API 可以在 Solaris、Windows NT 下运行,在将来的版本中 SDE 将对其它平台给予支持
 - 9) ARC/INFO 和 ArcView: ESRI 的 ARC/INFO GIS 和 ArcView 软件是 SDE 的首选客户机软件。SDE 与 ARC/INFO 软件间的转换,是在数据和系统水平上两个系统转向更加紧密耦合的第一步
7. **地理空間数据库引擎 ArcSDE:** ArcSDE 是 ArcGIS 与关系数据库之间的 GIS 通道, ArcSDE 为 DBMS 提供了一个开放的接口,允许 ArcGIS 在多种数据库平台上管理地理信息
8. **ArcSDE 可以提供两种模式:**
- 1) **B/S 结构下的 ArcSDE 工作流程:**

浏览器端: 提出数据申请或者发出交换数据请求

服务器端: ①对数据申请,则数据服务器(SDE)在 Oracle 数据库中执行检索及数据提取工作,并将目标数据经由数据缓冲池返回给浏览器端;②对交换数据的请求,服务器在综合数据库相应子库中开辟新的空间,将浏览器端传来的数据经由缓冲池存入,完成数据接收工作
 - 2) **C/S 结构下的 ArcSDE 工作流程:**

客户端: 向数据服务器提出数据申请

服务器: 根据客户端传来的参数在 Oracle 数据库中执行相应的空間搜索和数据提取工作,将满足搜索条件的目标数据存入数据缓冲池并发给应用服务器(SDE)或直接回传给客户端(视数据是否需要经由应用服务器进行预处理而定);

- ✧ **服务器和客户端异步协同工作：**服务器执行**所有**的空间查找和检索，并将结果返回给客户端。**一些耗费 CPU 资源较多的操作，如缓存计算、多边形覆盖，则在客户端运行。**

➤ Geodatabase

1. 是 ArcGIS 的核心数据模型，通过事务模型管理 GIS 数据 workflow。
2. 具有面向对象的特点：多态性、继承性、封装性
3. 空间数据与属性数据统一存储：一个实体=一个要素=一条记录
4. **可伸缩的存储解决方案——三种类型：**
 - 1) **个人地理数据库：**2GB 的存储上限，占用磁盘空间比较多
 - 2) **文件地理数据库：**单用户编辑，多用户浏览；以文件夹的形式表现，数据库无存储限制，每个表的存储上限为 1TB，相对于个人地理数据库占用磁盘空间比较小。
 - 3) **ARCSDE 地理数据库：**用户并发数无限制、存储于 RDBMS 中、支持版本管理、长事务 workflow。

第九章·空间数据处理与数据共享（数据获取）

1. **空间数据处理：**GIS 对空间数据本身的操作手段，不涉及内容的分析。
2. **空间数据处理的方法：**数据变换、数据重构、数据提取
3. **空间数据的变换：**空间数据坐标系的变换（实质：两个坐标系统下，点与点之间一一对应的关系），包括几何纠正和地图投影。
4. **空间数据结构的转换：**矢量向栅格的转换、栅格向矢量的转换。
5. **多元空间数据的融合：**遥感与 GIS 数据的融合、遥感影像与数字线划图 DLG 的融合、遥感影像与数字地形模型 DEM 的融合、遥感影像与数字栅格地图 DRG、不同 GIS 软件系统的空间数据格式的融合：（①基于转换器的数据融合（一般通过交换格式进行）。②基于直接访问的数据融合）
6. **空间数据的压缩和重分类：**①**压缩：**删除冗余数据，节省存储空间，以便后续处理；②**空间数据压缩：**基于矢量的压缩、基于栅格的压缩；③**重分类：**对属性的重分类
7. **空间数据的内插：**差值、逼近、拟合

第十章 · 空间数据库的发展

➤ 大数据与 NoSQL

1. **大数据**：大交易数据、大交互数据——二者融合是必然趋势
2. **大数据的“5V”特征**：①内容丰富、形式多样 (variety)；②容量巨大 (volume)；③数据的处理速度快 (velocity)；④数据的正确性 (veracity)；⑤数据价值高 (value)
3. **大数据时代下的系统需求**：①高并发读写的需求；②海量数据高效地存储和访问、查询；③需要拥有快速横向扩展能力以及 24 小时服务能力。
4. **关系数据库如何应对大数据？**
 - 1) **主从分离 (缓解写压力)**：主服务器将需要更新的信息写到特定的二进制文件，并自动把数据同步给从库。【主服务器是准实时的业务数据库，从服务器有延迟】
 - 2) **分库分表/数据切分 (缓解数据增长压力)**：把数据分散存储到多个数据库中。
 - 3) **增强读库的可扩展性**：
5. **什么是 NoSQL**：NoSQL 是 Not Only SQL 的缩写，它不一定遵循传统数据库的一些基本要求，相比传统数据库，叫它分布式数据管理系统更贴切。数据存储被简化，更灵活，重点被放在了分布式数据管理上。
6. **NoSQL 的两大理论基础**：①Google BigTable：把数据按列数据进行排序存储；②Amazon Dynamo：按 key 进行 hash 存储
7. **NoSQL 的优势**：①易拓展，数据之间无关系；②灵活的数据模型，随时可以存储自定义的数据格式；③在不太影响性能下可方便地实现高可用架构；④大数据量、高性能
8. **分布式数据系统的 CAP 原理**：在分布式系统中，一致性(Consistency)、可用性(Availability)、分区容忍性(Partition tolerance)这三个要素最多只能同时实现两点，不可能三者兼顾。【对于分布式数据系统，**分区容忍性是基本要求**；牺牲一致性而换取高可用性是目前多数分布式数据库产品的方向】
☆☆CAP 是 Nosql 不能替代其他关系数据库的原因☆☆
9. **NoSQL 与关系数据库的区别**：
 - 1) 关系型数据库是表格式的，容易关联协作存储，提取数据很方便；Nosql 数据库则与其相反，是大块的组合在一起。
 - 2) 关系型数据库对应的是结构化数据；Nosql 基于动态结构，使用非结构化数据。

- 3) 关系型数据库的数据存储为了更高的规范性，把数据分割为最小的关系表以避免重复；Nosql 数据存储在于平面数据集中，数据经常可能会重复，但整块数据更加便于读写。
 - 4) 关系型数据库是纵向扩展，也就是说想要提高处理能力，要使用速度更快的计算机；Nosql 数据库是横向扩展的，它的存储天然就是分布式的，可以通过给资源池添加更多的普通数据库服务器来分担负载。【最主要区别】
 - 5) 关系型数据库通过结构化查询语言（SQL）来操作数据库；Nosql 查询以块为单元操作数据，使用的是非结构化查询语言。
10. 主流的 NoSQL 数据库：①BigTable：分布式的结构化数据存储系统，是一个稀疏的、分布式的、持久化存储的多维度排序 Map，索引是行关键字、列关键字和时间戳。②Dynamo：P2P 的架构，数据定位使用一致性哈希。③Cassandra：开源的分布式 NoSQL 数据库系统，结合 BigTable 的数据模型和 Dynamo 系统架构，数据会写入多个节点。④HBase：高可靠性、高性能、面向列、可伸缩的分布式存储系统，是 BigTable 的开源实现。⑤Redis⑥MongoDB：基于分布式文件存储的数据库，介于非关系型数据库和关系型数据库的产品，支持的数据结构非常松散，支持的查询语言非常强大。不适用复杂的跨表查询。

➤ 空间数据挖掘

1. 空间数据仓库：是指支持管理的、决策过程的、面向主题的、集成的、随时间变化的、持久的和具有空间坐标的数据集合。
2. 空间数据挖掘（spatial data mining）：指从空间数据库中提取用户感兴趣的空间模式与特征、空间与非空间数据的普遍关系及其它一些隐含在数据库中的普遍的数据特征。也称为空间数据挖掘和知识发现（spatial data mining and knowledge discovery 简称为 SDMKD）
3. 数据挖掘技术的分类：
 - 1) 预言（Predication）：用历史预测未来
 - 2) 描述（Description）：了解数据中潜在的规律
4. 数据挖掘技术包括：关联分析、序列模式、分类（预言）、聚集、异常检测
5. 空间数据库的挖掘不同于经典的关系数据库的挖掘需求，特别是空间自相关（autocorrelation），即相似的对象趋于在地理空间中进行聚集。

6. **数据挖掘的过程的重要子过程有：**数据提取、数据清理、特征选取、算法设计和调整以及当算法应用与特定数据时所得到的输出结果进行分析。
7. **空间数据挖掘的任务概括如下：**在**空间数据库和数据仓库**的基础上，**综合利用**多领域**相关的信息技术手段**，从大量的空间数据、管理数据、经营数据或遥感数据中**析取**知识，从而揭示出蕴含在空间数据背后客观世界的本质规律、内在联系和发展趋势，实现知识的自动或半自动获取，**为决策提供依据**。